



UniCEUB – Centro Universitário de Brasília  
FAET – Faculdade de Ciências Exatas e Tecnologia  
Curso de Engenharia da Computação

# **Reconhecimento de Caracteres Manuscritos Utilizando Redes Neurais Artificiais**

Aluno: Carlos Eduardo Sousa de Carvalho – R.A.: 997277/2

Orientador: Prof. MSc. Aderlon M. Queiroz

Brasília - 2006



UniCEUB – Centro Universitário de Brasília  
FAET – Faculdade de Ciências Exatas e Tecnologia  
Curso de Engenharia da Computação

# **Reconhecimento de Caracteres Manuscritos Utilizando Redes Neurais Artificiais**

Monografia, sob a orientação do Prof. MSc.  
Aderlon Marcelino Queiroz avaliado por Banca  
Examinadora do Curso de Engenharia da  
Computação da Faculdade de Ciências Exatas e  
Tecnologia - FAET do Centro Universitário de Brasília  
- UniCEUB e constitui requisito para obtenção do  
título de Bacharel em Engenharia da Computação.

Brasília – DF, novembro de 2006.

## Agradecimentos

A Deus, pelo dom da vida.

Aos meus pais, pela dedicação e amor depositados em mim todos estes anos.

Aos meus irmãos, pelo incentivo no decorrer desta graduação.

A minha amada Danila, pelo carinho, afeto e companheirismo.

Ao professor da disciplina de Projeto Final, MSc. Francisco Javier, por se dedicar intensivamente a seus alunos sempre com muita paciência.

Ao meu orientador, Prof. MSc. Aderlon M. Queiroz, pela enorme ajuda prestada na realização desse projeto.

## Resumo

Este projeto consiste em desenvolver um sistema capaz de reconhecer vogais manuscritas digitalizadas em formato *bitmap*, podendo ser posteriormente utilizado para integrar projetos de natureza mais complexas, como o radar inteligente.

Destaca-se nesse projeto a utilização de Redes Neurais Artificiais para que se possa generalizar as imagens a serem reconhecidas. Sendo que esse projeto foi desenvolvido em linguagem C, visando portabilidade.

**Palavras-Chaves:** Redes Neurais, Reconhecimento de Caracteres, Linguagem C, Vogais.

## Abstract

This project consists on developing a system in language C, which provides portability, capable to recognize vowels written by hand in bitmap format, and it can be used to integrate other complex projects, as the intelligent radar.

The use of Artificial Neural Nets is distinguished in this project because it can generalize the images to be recognized.

**Keywords:** neural networks, patten recognition, language C, vowels.

# Sumário

Lista de Figuras.....	VIII
Lista de Tabelas.....	IX
Capítulo 1 – Introdução.....	1
1.1 Motivação.....	1
1.2 Objetivo.....	2
1.3 Estrutura da Monografia.....	2
Capítulo 2 – Reconhecimento de Padrões.....	3
2.1 OCR.....	4
Capítulo 3 – Redes Neurais Artificiais.....	6
3.1 O que são Redes Neurais Artificiais?.....	6
3.2 Histórico.....	7
3.3 Conceitos Básicos.....	9
3.3.1 Neurônio Biológico.....	10
3.3.2 Potencial de Ação.....	12
3.3.3 Função de Ativação.....	13
3.3.4 Aprendizado.....	15
3.3.4.1 Aprendizado Supervisionado.....	16
3.3.4.2 Aprendizado Não-Supervisionado.....	17
3.4 Perceptron.....	18
3.5 Adaline.....	22
3.6 MLP.....	22
3.6.1 Back-propagation.....	23
Capítulo 4 – Protótipo.....	27
4.1 Ferramenta Utilizada.....	27
4.2 Métodos e Recursos Utilizados.....	27
4.3 Distribuição do Desenvolvimento do Projeto.....	27
4.3.1 Disposição dos Arquivos.....	28
4.3.2 Vetorização da Imagem.....	31

4.3.3 Geração dos Pesos.....	31
4.3.4 Rede Neural Artificial.....	32
Capítulo 5 – Simulações e Resultados.....	34
5.1 Execução dos Programas.....	35
5.1.1 Disposição da Ordem das Imagens.....	35
5.1.2 Vetorização das Imagens.....	36
5.1.3 Geração dos Pesos.....	37
5.1.4 Rede Neural Artificial.....	39
5.2 Resultados.....	43
5.2.1 Sedna utilizando $\eta = 0,3$ .....	48
5.2.2 Plutão utilizando $\eta = 0,3$ .....	53
5.2.3 Sedna utilizando $\eta = 0,45$ .....	58
5.2.4 Plutão utilizando $\eta = 0,45$ .....	62
5.2.5 Arquivos diversos utilizando $\eta = 0,45$ .....	65
5.2.6 Arquivos diversos utilizando $\eta = 0,3$ .....	67
Capítulo 6 – Conclusão.....	69
6.1 Considerações Finais.....	69
6.2 Dificuldades Encontradas.....	70
6.3 Sugestão para Trabalhos Futuros.....	70
Referências.....	71
Anexo A – Algoritmos.....	72
A.1 Sedna.....	72
A.2 Plutão.....	86
A.3 Netuno.....	103
A.4 Urano.....	119
A.5 Saturno0.....	135
A.6 Saturno1.....	152
Anexo B – Resultados.....	167

## Lista de Figuras

Figura 3.1 – Neurônio do sistema nervoso central.....	11
Figura 3.2 – Potencial de Ação.....	13
Figura 3.3 – Funções de ativação.....	14
Figura 3.4 – Aprendizado Supervisionado.....	16
Figura 3.5 – Aprendizado não-supervisionado.....	17
Figura 3.6 – Perceptron de Camada simples.....	18
Figura 3.7 – Fluxo do Sinal do Perceptron.....	19
Figura 3.8 – Hiperplano com fronteira de Decisão.....	21
Figura 3.9 – Rede MLP com uma camada oculta.....	23
Figura 3.10 – Fluxo do processamento do algoritmo back-propagation.....	24
Figura 5.1 – Fluxo dos programas.....	34
Figura 5.2 – Execução do programa Sedna.....	35
Figura 5.3 – Execução do programa Plutão.....	36
Figura 5.4 – Execução do programa Netuno.....	37
Figura 5.5 – Execução do programa Urano.....	38
Figura 5.6 – Execução da primeira parte do programa Saturno0.....	39
Figura 5.7 – Continuação da execução do programa Saturno0.....	41
Figura 5.8 – Execução da primeira tela do programa Saturno1.....	42
Figura 5.9 – Continuação da execução do programa Saturno1.....	43
Figura 5.10 – Ocorrência de OVERFLOW no programa Saturno1.....	44
Figura 5.11 – Início da evolução do reconhecimento.....	45
Figura 5.12 – Metade da evolução do reconhecimento.....	46
Figura 5.13 – Fim da evolução do reconhecimento.....	47



## Lista de Tabelas

Tabela 4.1 – Amostra da distribuição gerada pelo Sedna.....	27
Tabela 4.2 – Representação das vogais.....	27
Tabela 4.3 – Amostra da distribuição gerada pelo Plutão.....	30
Tabela 5.1 – Amostra da lista que será processada.....	40
Tabela 5.2 – Resultado dos arquivos 3x1 do Sedna com $\eta = 0,3$ .....	48
Tabela 5.3 – Tabela 5.3 Distribuição dos arquivos que utilizam o agrupamento 3x1. .....	48
Tabela 5.4 – Resultado dos arquivos 2x2 do Sedna, com $\eta = 0,3$ .....	50
Tabela 5.5 – Distribuição dos arquivos que utilizam o agrupamento 2x2.....	50
Tabela 5.6 – Resultado dos arquivos 1x3 do Sedna, com $\eta = 0,3$ .....	52
Tabela 5.7 – Distribuição dos arquivos que utilizam o agrupamento 1x3.....	52
Tabela 5.8 – Resultado dos arquivos 3x1 do Plutão, com $\eta = 0,3$ .....	54
Tabela 5.9 – Resultado dos arquivos 2x2 do Plutão, com $\eta = 0,3$ .....	55
Tabela 5.10 – Resultado dos arquivos 1x3 do Plutão, com $\eta = 0,3$ .....	56
Tabela 5.11 – Resultado dos arquivos 3x1 do Sedna, com $\eta = 0,45$ .....	59
Tabela 5.12 – Resultado dos arquivos 2x2 do Sedna, com $\eta = 0,45$ .....	60
Tabela 5.13 – Resultado dos arquivos 1x3 do Sedna, com $\eta = 0,45$ .....	61
Tabela 5.14 – Resultado dos arquivos 3x1 do Plutão, com $\eta = 0,45$ .....	62
Tabela 5.15 – Resultado dos arquivos 2x2 do Plutão, com $\eta = 0,45$ .....	63
Tabela 5.16 – Resultado dos arquivos 1x3 do Plutão, com $\eta = 0,45$ .....	64
Tabela 5.17 – Resultado para 2000 imagens com $\eta = 0,45$ .....	66
Tabela 5.18 – Ordem de gravação do arquivo Quaoar.....	67
Tabela 5.19 – Resultado para 2000 imagens com $\eta = 0,3$ .....	67
Tabela 5.20 – Ordem de gravação do arquivo Sedna.....	68

## Capítulo 1 – Introdução

### 1.1 Motivação

O reconhecimento de padrões tem sido cada vez mais estudado e aplicados nos últimos anos. É cada vez maior o número de sistemas que tendem substituir o ser humano em tarefas que não necessitam de serem tomadas decisões diferentes das pré-estabelecidas, ou seja, quando se é utilizado o reconhecimento de padrões para verificar o acesso de uma pessoa a um determinado lugar, só existem duas respostas, acesso autorizado, ou acesso negado.

Outro exemplo em que há a substituição do homem por um sistema inteligente, é o uso do radar inteligente, onde antes era necessário que o agente fiscalizador solicitasse que o motorista pare o veículo, depois com posse do número da placa, verificasse a situação do veículo junto a um computador interligado ao sistema do DETRAN, para verificar se há alguma restrição sobre o veículo. Isso demorava um certo tempo, que com a utilização do radar inteligente, a verificação é feita em frações de segundos e para todos os veículos que passarem em frente ao radar, e não mais por amostragem, como era anteriormente.

Outro conceito que vem crescendo amplamente são as Redes Neurais Artificiais, que faz o computador agir de forma semelhante ao comportamento do cérebro humano. Com o uso da rede neural, o computador é capaz de realizar tarefas que até então era impossível, tais como, aprender através de exemplos e generalizar as informações aprendidas.

## 1.2 Objetivo

Desenvolver um software aplicativo, que seja capaz de reconhecer uma determinada imagem no formato *bitmap*, sendo esta imagem uma vogal maiúscula ou minúscula, onde será utilizado o conceito de redes neurais artificiais.

Serão utilizadas para reconhecimento as vogais devido a estas serem um número menor de possibilidades, ou seja, apenas 5 de um total de 26 do alfabeto ocidental.

Outro objetivo é fazer uma análise sobre o comportamento dos resultados obtidos como a execução do aplicativo.

## 1.3 Estrutura da monografia

Será descrita brevemente a composição do trabalho em cada capítulo:

O capítulo 2 tem como intenção fazer uma breve abordagem sobre o conceito de reconhecimento de padrões, além de citar o OCR.

No capítulo 3 são abordadas as Redes Neurais Artificiais, com histórico, definições, características, conceitos básicos, além dos tipos de redes.

No capítulo 4 é apresentado o protótipo, explicando como ficaram distribuídos os módulos que o compõe, explicando o funcionamento em geral.

O capítulo 5 mostra os resultados obtidos na realização das simulações e testes, além de serem feitas algumas considerações sobre estes resultados.

E por último, no capítulo 6, é apresentada a conclusão do projeto, fazendo as considerações finais, informando as dificuldades encontradas e propondo trabalhos futuros.

## Capítulo 2 – Reconhecimento de Padrões

Reconhecer é uma importante tarefa do dia a dia, sendo uma propriedade básica dos seres humanos. Quando alguém observa um objeto, primeiramente reconhece todas as características que possa perceber, depois compara essas características com algo parecido que está armazenado na mente. Se há semelhança entre o que está se vendo e o que está armazenado no conhecimento, o objeto é reconhecido.

Essa tarefa de reconhecer é um tanto simples e familiar no mundo real, porém no mundo da inteligência artificial é um grande feito, isso porque o cérebro humano não se compara a nenhuma máquina ou software artificiais.

O ato de reconhecer pode ser dividido em duas categorias: reconhecer algo concreto e reconhecer algo abstrato. Para o reconhecimento concreto, pode se ter como exemplo um vaso, uma assinatura, ou até mesmo uma forma de onda, ou seja, apesar de não podermos sentir com o tato, estes exemplos existem e não necessitam do uso da imaginação para ser deduzido. O reconhecimento abstrato é atribuído a coisas que não existem fisicamente. Sendo assim será tratado apenas o reconhecimento concreto. [JPJ 06]

O reconhecimento de padrões envolve três processos: entrada filtrada, extração das características e classificação.

- Entrada filtrada está relacionada a retirada das informações que não são pertinentes ao contexto, ou seja, retirar da entrada o que não é desejado. Por exemplo, em um reconhecimento de voz deve-se tentar retirar o ruído.
- A extração da característica é um processo de estudo e derivação das informações úteis da entrada que foi filtrada. As informações derivadas podem ser as características gerais, que são avaliadas para facilitar o processo seguinte. Por exemplo, no reconhecimento de imagens, as características extraídas conterão a informação sobre a máscara cinzenta, a textura, a forma ou o contexto da imagem. Estes métodos de extração são dependentes da aplicação utilizada.

- A classificação é o estágio final do teste de reconhecimento de padrões, é onde o sistema automatizado declara que o objeto de entrada é pertencente a uma determinada categoria particular.

Os problemas encontrados para o reconhecimento de imagens são três: escala, rotação, translação e perspectiva.

- O problema de escala: verificado quando o tamanho da imagem a ser reconhecida está reduzido ou aumentado em relação à imagem treinada.
- O problema de translação: verificado quando a imagem a ser reconhecida está com uma inclinação diferente da imagem treinada.
- O problema de rotação: verificado quando a imagem a ser reconhecida está em uma posição diferente da imagem treinada.
- O problema de perspectiva: verificado quando a imagem a ser reconhecida está com um ângulo diferente da imagem treinada.

## 2.1 OCR

OCR é uma sigla de *Optical Character Recognition*, ou melhor, Reconhecimento Óptico de Caractere, uma tecnologia para reconhecer caracteres a partir de uma lista de imagens.

No início dos anos 50 David Shepard e Dr. Louis Tordella iniciaram pesquisas a respeito de automação de dados da Agência de Segurança dos Estados Unidos, mas foi com a ajuda de Harvey Cook que eles conseguiram desenvolver o primeiro software relacionado a reconhecimento óptico de caracteres, sendo este software chamado de “GISMO”. Depois, David Shepard fundou a Intelligent Machines Research Corporation (IMR) onde foi desenvolvido os primeiros softwares voltados para o mercado. E em 1953, a IBM obteve uma licença junto a IMR, para desenvolver um software próprio,

sendo chamado de Optical Character Recognition (OCR), tornando-se um padrão dessa tecnologia.

Os primeiros sistemas comerciais só reconheciam dois tipos de fontes além de estarem apresentadas em um formato bem comportado como um livro. Hoje em dia são capazes de reconhecer qualquer tipo de fonte. [NCE 06]

O OCR detêm uma tecnologia muito rígida em relação a variações dos caracteres, caso o caractere que está sendo verificado não seja dado como idêntico aos caracteres que ele possui em sua base, o OCR não irá reconhecer este caractere. Através do OCR é possível digitalizar um papel com um texto e depois utilizá-lo para ser editado.

Outro exemplo de tecnologia que utiliza o reconhecimento de caracteres é o ICR (*Intelligent Character Recognition*, ou Reconhecimento de Caracteres Inteligente), que é uma versão avançada do OCR. O ICR é capaz de lidar com problemas bem mais complexos que o OCR, uma vez que os caracteres reconhecidos não são necessariamente impressos, podendo ser até mesmo manuscritos. Um exemplo de aplicação onde pode ser encontrado o ICR é o palm-top.

## Capítulo 3 – Redes Neurais Artificiais

### 3.1 O que são Redes Neurais Artificiais?

A partir de estudos feitos sobre o funcionamento do cérebro humano, foi verificado que o processamento deste é completamente diferente do computador convencional. Pode-se, com isso, constatar que o cérebro humano é um computador altamente complexo que pode processar tarefas extremamente difíceis em frações de segundos, ao passo que um computador convencional pode levar várias horas.

As Redes Neurais Artificiais (RNAs) constituem um método de solucionar problemas que se originam da Inteligência Artificial (IA), construindo sistemas que tenham circuitos que simulem o cérebro humano, ou seja, este circuito aprende, erra, e até mesmo faz algum tipo de descoberta. Em outras palavras, são técnicas computacionais que apresentam um modelo inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento com a experiência. Uma grande Rede Neural Artificial (RNA) pode chegar a centenas ou até mesmo milhares de unidades de processamento, enquanto isso, o cérebro humano tem alguns bilhões de neurônios.

A IA pode ser definida como uma área da computação. Ela reúne uma série de técnicas e metodologias de programação que são usadas para tentar resolver os problemas onde se necessita uma certa 'inteligência'. Essas técnicas e metodologias tendem a ser mais eficientes que soluções algorítmicas convencionais e fazem isso o mais próximo possível do processamento do cérebro humano. Ou seja, o principal objetivo da IA é tentar fazer que um sistema consiga processar informações da mesma maneira que um ser humano processa as suas.

Uma RNA pode ser implementada tanto em hardwares quanto em softwares. Nela são utilizados vários neurônios artificiais simples com conexões entre si. Devido a essas interligações, as RNAs também são conhecidas como connexionismo. Através dessas conexões, a rede recebe informações de sensores ou de outros neurônios artificiais produzindo operações simples sobre estes dados e passando o resultado para

outros neurônios artificiais. As RNAs funcionam por meio de neurônios artificiais, que processam os dados usando paralelismo lógico (para todos os neurônios da mesma camada) e combinado com operações seriais (quando a informação de uma camada é transferida para neurônios de outra camada). Existem três características principais que descrevem uma rede neural e que contribuem para a sua habilidade funcional: estrutura (ou arquitetura), dinâmica e aprendizado.

A rede neural existe primeiramente a fase de aprendizagem, em que as amostras são inseridas na rede, que retira as características necessárias para representar a informação recebida. Essas características serão posteriormente utilizadas para a resolução do problema.

Redes Neurais constituem uma tecnologia computacional emergente que pode aumentar significativamente o número de aplicações. As RNAs estão sendo adotadas para o uso em uma variedade de aplicações comerciais e militares que atingem o reconhecimento de padrões até a otimização e seqüenciamento. São desenvolvidas em um tempo razoável e, normalmente, realizam as tarefas bem melhor que as outras tecnologias mais convencionais (incluindo sistemas especialistas). [LOE96]

## 3.2 Histórico

Em 1943, o psiquiatra e neuroanatomista Warren McCulloch juntamente com o matemático Walter Pitts realizaram um trabalho pioneiro sobre RNAs. Eles publicaram um artigo com o título: "*A Logical Calculus of the Ideas Immanent in Nervous Activity*" (Um Cálculo das Idéias Imanentes em Atividade Nervosa). O modelo de McCulloch e Pitts (ou modelo MCP) é baseado na atribuição de um neurônio à função de soma e *threshold*, onde os pesos nas conexões entre os nodos correspondem às sinapses inibidoras de um neurônio real. Embora não esteja tão claro no trabalho original de McCulloch e Pitts, os nodos MCP são capazes de se adaptarem por meio do ajuste dos pesos de cada conexão. Esse processo de adaptação permite o desenvolvimento de procedimento para o ajuste gradual dos pesos, de forma que a rede desempenhe a função desejada. [BCL 00]



Depois do artigo de McCulloch e Pitts nenhum trabalho acrescentou muito, até que em 1949, Donald Hebb publicou o livro "*The Organization of Behavior*" (A Organização do Comportamento), que resultou no primeiro trabalho sobre o aprendizado. Ele mostrou como a capacidade da rede neural pode ser modificada e conseguida a partir da variação dos pesos de entrada dos nodos. Hebb propôs uma teoria para explicar o aprendizado em nodos biológicos baseada nas ligações sinápticas entre os nodos excitados. A regra de Hebb é utilizada hoje em vários algoritmos de aprendizado. Mais tarde, Widrow e Hoff sugeriram uma regra de aprendizado, conhecida como regra Widrow-Hoff, ou regra Delta, que é baseada no cálculo do erro entre a soma atual dos pesos das entradas e a saída desejada, ajustando o peso de forma que o erro se torne zero. [LF 99 apud GRU 99]

Em 1958, quinze anos depois dos pioneiros McCulloch e Pitts publicarem o clássico artigo, Frank Rosenblatt publicou uma nova abordagem sobre o problema de reconhecimento de padrões, o Perceptron, um método inovador de aprendizado. O modelo de rede de Rosenblatt, baseado no nodo MCP, era capaz de classificar os padrões aprendidos do ambiente por meio do ajuste gradual dos pesos entre os nodos. Rosenblatt defendia a idéia que o cérebro humano trabalhava como associador de padrões adaptáveis e não como um circuito lógico determinístico como pensavam Minsky e von Neumann. [BCL 00]

No entanto, os Perceptrons tiveram uma vida curta, pois, até então, se achava que as redes neurais poderiam resolver qualquer coisa. E, em 1969, Marvin Minsky, um dos adeptos da visão criticada por Rosenblatt, publicou um livro, demonstrando que existem limites para o Perceptron de uma única camada. O Perceptron não era capaz de resolver problemas linearmente separáveis. [BCL 00]

Na década de 70, com a repercussão do trabalho de Minsky e Papert, as pesquisas referentes a redes neurais ficaram um pouco de lado. Com isso, foram poucos pesquisadores que continuaram seus projetos nesse ramo.

Em 1982, houve uma retomada dos trabalhos sobre redes neurais devido à criação do Modelo de Hopfield, produzido pelo físico John Hopfield. Este modelo se caracteriza por ser do tipo feedback, isto é, há uma conexão das entradas com as

saídas. Com isso, vai haver um momento em que a saída, após oscilar entre alguns valores, chegará sempre a mesma resposta para o mesmo padrão de entrada.

Logo depois, surgiram outros trabalhos que ajudaram a gerar uma explosão de interesse pelas RNAs, tais como, o desenvolvimento de uma máquina estocástica conhecida como Máquina de Boltzmann, que foi a primeira obra de sucesso que utilizou uma rede neural de múltiplas camadas. Outro trabalho foi a descrição do algoritmo de treinamento de back-propagation, onde mostrou que a RNA é uma área promissora. [HAY 01]

Com o avanço nas pesquisas a partir da década de 80, houve uma grande procura de informação sobre as RNAs pela comunidade internacional. Isso ocorreu devido às últimas descobertas, onde ficou clara a pontencialidade das mesmas.

Sistemas neurais sem peso também tiveram sua importância na história da RNA, pois proporcionaram modelos de rápido aprendizado e fácil implementação. A visão não convencional das Redes Neurais Sem Peso (RNSP) consiste em armazenar a função do nodo em uma memória RAM ao invés de ajustar os pesos entre os nodos. Essa abordagem facilita o desenvolvimento de algoritmos de aprendizado mais simples do que os baseados no modelo MCP, uma vez que existe uma independência entre os nodos durante o processo de aprendizagem. O modelo sem peso mais conhecido é certamente o WISARD (Aleksander, Thomas e Bowden, 1984), o qual se tornou disponível como uma máquina comercial no início dos anos 80, estando ainda em uso nas aplicações de reconhecimento de padrões.

### 3.3 Conceitos Básicos

A seguir serão postados alguns conceitos importantes para se entender o funcionamento da rede neural artificial.

### 3.3.1 Neurônio Biológico

O cérebro é caracterizado por executar um grande número de tarefas, mas quem realiza todo esse procedimento são os neurônios biológicos que são interligados uns aos outros. O neurônio é constituído por uma membrana celular fina que além de sua função biológica tem grande importância sobre os estímulos elétricos da célula nervosa. Ele é composto pelo corpo celular, onde são processadas as informações da célula, e por extensões filamentosas longas que terminam em inúmeras e finas ramificações, conectadas a outro neurônio vizinho, em locais chamados sinapses. Os filamentos são de dois tipos, o dentrito e o axônio. Os dentritos se iniciam em qualquer parte da superfície do corpo celular e se assemelham muito aos ramos de uma árvore. E o axônio, que também é conhecido como fibra nervosa, é caracterizado na maioria dos casos por ser único e não variar o seu diâmetro. [KOV 96]

A célula nervosa deve receber as variações do ambiente em uma extremidade e conduzi-la até a outra extremidade. Na parte mais extrema do neurônio está o dentrito que terá a função de receber as variações do ambiente que é apenas um estímulo. O efeito desse estímulo dentro do neurônio é o de excitá-lo. Essa excitação resultante, que é a mensagem ou o sinal nervoso será entregue ao axônio.

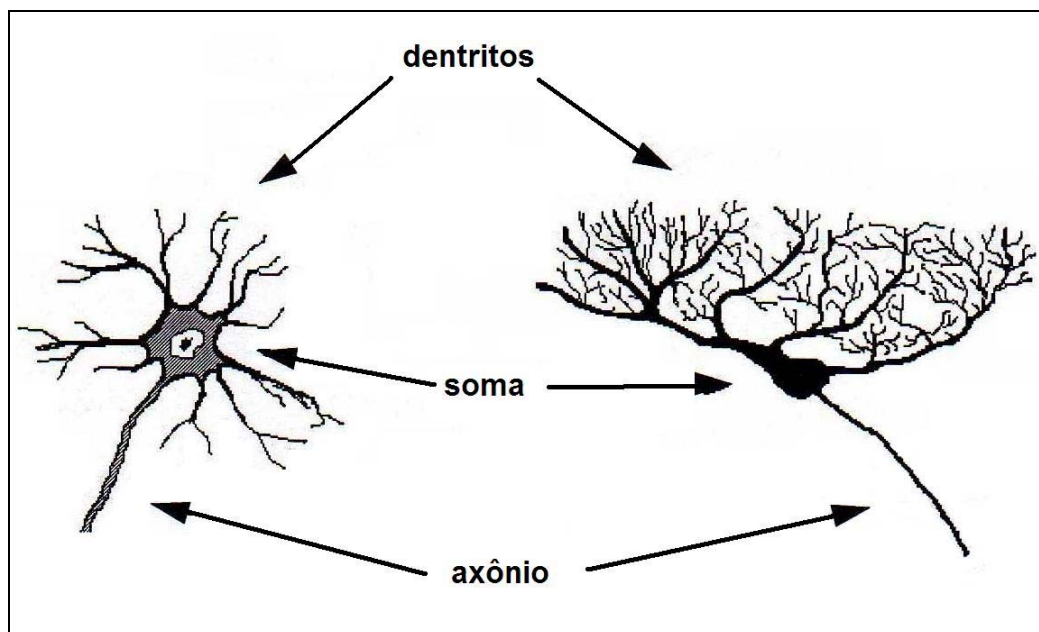


Figura 3.1 Neurônio do sistema nervoso central

Fonte: Kovács

O neurônio é a célula que mais se diferencia das outras células do organismo, pois nenhuma é tão complexa quanto ele em relação a sua estrutura e funcionalidade. O neurônio é formado na vida embrionária do bebê, onde sofrem as divisões celulares. Após o nascimento, a quantidade desta célula não se altera até o fim da vida, o que ocorre é o crescimento das ramificações e conexões com outras células.

Os principais componentes do neurônio são:

O corpo de neurônio, também conhecido como soma – responsável por coletar e combinar informações de outros neurônios.

Os dentritos – recebem os estímulos transmitidos por outros neurônios.

O axônio – responsável por transmitir os estímulos a outros neurônios, é constituído por uma fibra tubular que pode chegar a vários metros.

Foi constatado pela primeira vez por DuBois Reymond manifestações elétricas nos neurônios biológicos no século XIX com o auxílio de galvanômetros. Décadas mais tarde, os pesquisadores consideraram que o neurônio era simplesmente um dispositivo

computacional elementar do sistema nervoso, que possuía entradas e uma saída. Os sinais que chegam nos dendritos são impulsos nervosos ou potenciais de ação, e são basicamente as informações que o neurônio processará. [KOV 96]

A conexão entre os dendritos de uma célula ao axônio de outra é dada a partir das conexões sinápticas. As sinapses são regiões eletroquimicamente ativas, compreendidas entre duas membranas celulares. As conexões poderão ser de duas formas: excitatória ou inibitória. “A conexão excitatória altera o potencial da membrana que contribuiu para a formação de um impulso nervoso no axônio de saída enquanto que a conexão inibitória age no sentido contrário”. [KOV 96 pág 15]

### 3.3.2 Potencial de Ação

Dentro do axônio existe uma membrana que quando está em repouso, ou seja, sem a influência de um impulso nervoso, está a um potencial eletronegativo de algumas dezenas de milivolts em relação ao exterior da membrana. “Essa diferença de potencial é sustentada por um processo de difusão assimétrica de íons de sódio e potássio através da membrana, processo este conhecido como bomba de sódio”. Existem outros dois estados em que a membrana interior pode se encontrar: despolarizada ou hiperpolarizada. O estado despolarizada é quando a membrana está menos eletronegativa em relação ao estado de repouso e quando está mais eletronegativo considera-se que ela está hiperpolarizada. [KOV 96]

O potencial de ação é descrito da seguinte forma: primeiro a membrana fica muito despolarizada, chegando a um valor que é chamado de limiar de disparo. Depois, através de um princípio ativo a membrana cessa o estado de polarização rapidamente e em seguida utilizando um período bem maior, volta ao estado de repouso. [KOV 96]

“Após a ocorrência de um impulso nervoso, a membrana entra em um período conhecido como período de refração absoluta, durante o qual é incapaz de produzir um outro potencial de ação, independente da intensidade da despolarização. Seguido desse período de refração absoluta, persiste um período de refração relativa, correspondendo a uma elevação no limiar de disparo que, assintoticamente, retorna ao

seu valor normal. Durante o período de refração relativa, a fibra é capaz de produzir potencial de ação, porém com depolarizações mais intensas”. [KOV 96 pág 17]

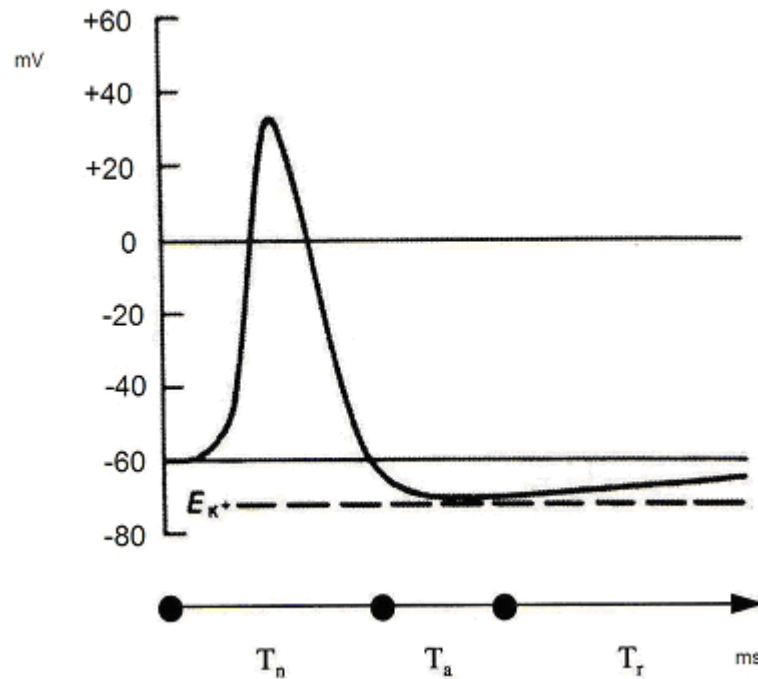


Figura 3.2 Potencial de Ação representado pelo potencial de membrana no axônio de um neurônio típico. Onde  **$T_n$**  é a duração do impulso nervoso,  **$T_a$**  é o período de refração absoluta e  **$t_r$**  o período de refração relativa.

Fonte: Kovács

### 3.3.3 Função de Ativação

A função de ativação é localizada após serem somados as entradas ponderadas pelos respectivos pesos sinápticos. Ela tem a função de restringir a amplitude da saída do neurônio a um valor finito, por essa característica, a função de ativação é também conhecida como função restritiva. [HAY 01]

A figura a seguir mostra algumas das funções de ativação.

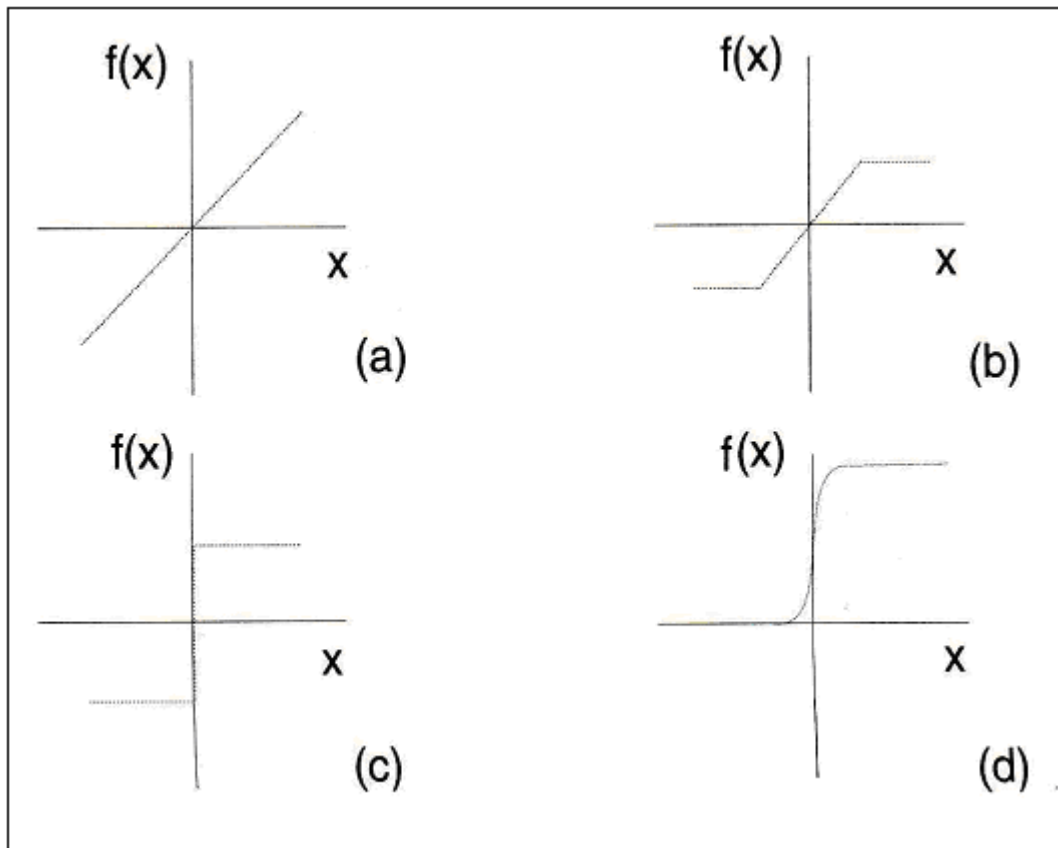


Figura 3.3 Funções de ativação

Fonte: Braga

Função de ativação linear, representada pela figura 3.3a

$$f(x) = x \quad (3.1)$$

Função de ativação rampa, representada pela figura 3.3b.

$$f(x) = \begin{cases} +\gamma, & x \geq +\gamma \\ x, & |x| < +\gamma \\ -\gamma, & x \leq -\gamma \end{cases} \quad (3.2)$$

Onde,  $+\gamma$  e  $-\gamma$  representam os valores máximo e mínimo.

Função de ativação degrau simétrico, representada pela figura 3.3c

$$f(x) = \begin{cases} 1, x > 0 \\ -1, x < 0 \end{cases} \quad (3.3)$$

Função de ativação sigmoideal representada pela figura 3.3d

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3.4)$$

### 3.3.4 Aprendizado

A principal característica da rede neural é a capacidade de aprender a partir do seu ambiente, essa capacidade é adquirida com um processo de aprendizagem. O aprendizado fornece subsídios necessários para que a rede neural possa melhorar o seu desempenho com o passar do tempo. A etapa de aprendizado é realizada a partir do ajuste dos pesos que são utilizados em cada conexão, esse ajuste é feito a todo o instante.

Uma definição interessante sobre o processo de aprendizado é:

“Aprendizagem é o processo pelo qual os parâmetros de uma rede neural são ajustados através de uma forma continuada de estímulo pelo ambiente no qual a rede está operando, sendo o tipo específico de aprendizagem realizada definido pela maneira particular como ocorrem os ajustes nos parâmetros”. [BCL 00 appud MM 70]

Existem dois métodos principais de treinamento que podem ser atribuídos ao aprendizado: aprendizado supervisionado que será aplicado neste projeto e aprendizado não-supervisionado.



### 3.3.4.1 Aprendizado Supervisionado

O método de aprendizado supervisionado é o mais comum entre os treinamentos de RNAs. Ele também é conhecido como treinamento com professor, pois a entrada e a saída são fornecidas por um supervisor externo. A figura abaixo mostra um diagrama de blocos onde mostra o comportamento do professor. [BCL 00]

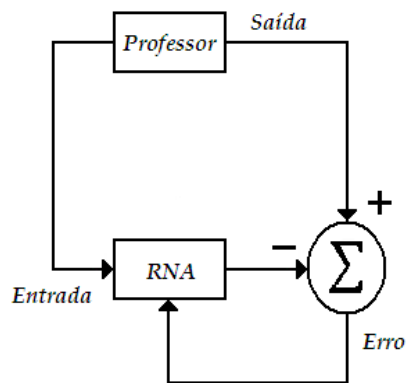


Figura 3.4 Aprendizado Supervisionado

Fonte: Braga

Como o professor indica a entrada, ele sabe qual a saída desejada. Dessa forma o professor sabe como devem ser ajustados os parâmetros para que a saída possa ser corrigida. A intenção é encontrar uma ligação entre a entrada e a saída.

Para se efetuar um treinamento, é necessário ter um conjunto de entradas a serem treinadas e um conjunto de saídas desejadas para cada elemento da entrada. Com isso, toda vez que um novo elemento da entrada é processado pela RNA, o professor já tendo conhecimento da saída desejada, compara a saída obtida com a desejada. Caso as saídas não coincidirem, os pesos são ajustados na intenção de minimizar o erro da rede. O ajuste dos pesos é feito de forma continuada para cada nova entrada, fazendo o erro tender a zero.

A desvantagem que se pode ter com a utilização do treinamento supervisionado é que se houver uma ausência do professor, a rede não poderá desenvolver uma nova estratégia para os casos onde o professor não atenda com os exemplos.

A correção do erro é dada pela seguinte fórmula abaixo:

$$w_i(t+1) = w_i(t) + \eta e(t) x_i(t) \quad (3.5)$$

Onde:  $\eta$  é a taxa de aprendizado.

$x_i(t)$  é a entrada de neurônio  $i$  no tempo  $t$ .

### 3.3.4.2 Aprendizado Não-Supervisionado

No aprendizado não-supervisionado não há existência de nenhum professor que possa controlar os resultados e se assemelha a alguns sistemas como, por exemplo, a visão e a audição em seus estágios iniciais. Este tipo de aprendizado não requer saídas desejadas, por isso não necessita de nenhum tipo de professor. O aprendizado não-supervisionado trabalha apenas com elementos de entrada e cria um grupo de elementos a partir de seu histórico, cada entrada é classificada de acordo com os próprios elementos da entrada. Utiliza os elementos como classificadores e os elementos como a classificar; esta classificação fica a critério da rede e do algoritmo de aprendizado. Este aprendizado tem como característica a grande capacidade de aprender, isso porque consegue discriminar cada estímulo de cada elemento apresentado.

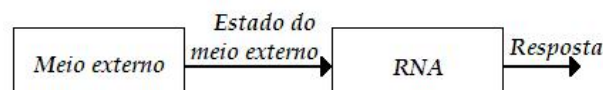


Figura 3.5 Aprendizado não-supervisionado

Fonte: Braga

### 3.4 Perceptron

Em 1958, Rosenblatt demonstrou algumas aplicações práticas usando Perceptron. “O Perceptron é a forma mais simples de uma rede neural usada para a classificação de padrão”. Ele é capaz de separar linearmente vetores de entradas em classes de padrões através de hiperplanos. [HAY 01]

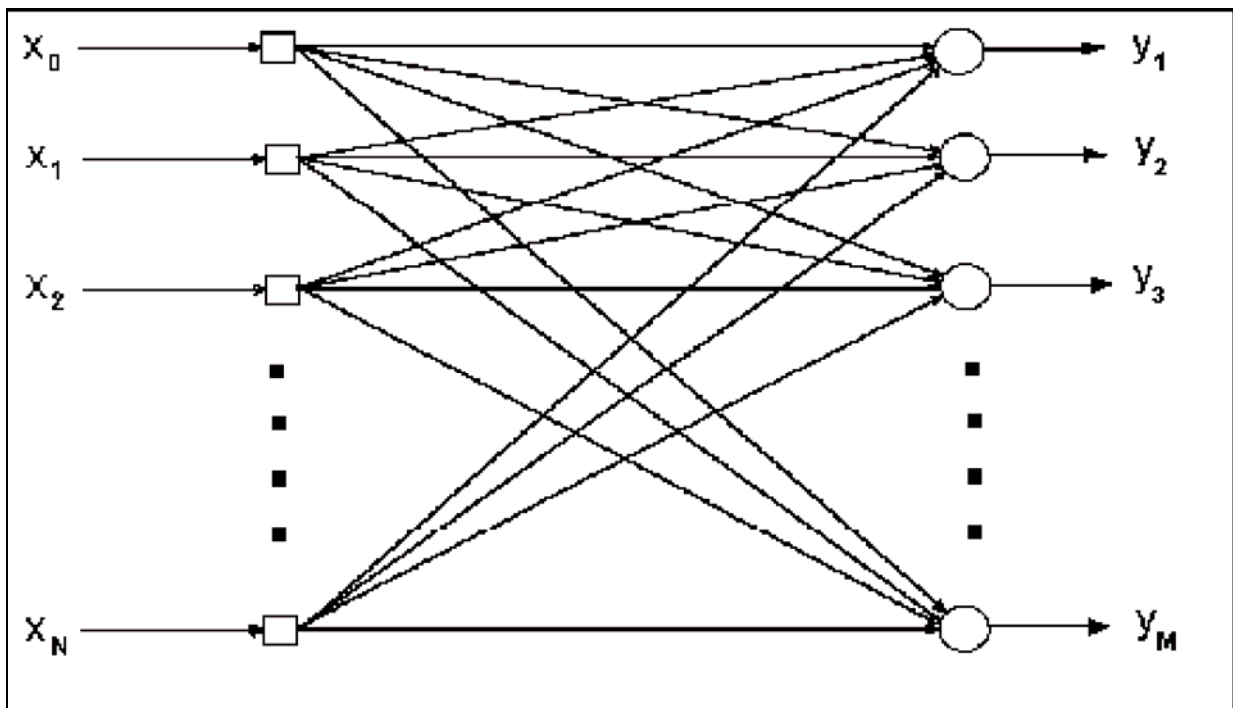


Figura 3.6 Perceptron de Camada simples.

Fonte: Haykin

A figura anterior apresenta o Perceptron de N-entradas e M-saídas. Este modelo pode ser descrito por:

$$Y_i = f \left\{ \sum_{j=0}^n w_{ij} x_j \right\} \quad i=1, 2, \dots, M \quad (3.6)$$

Onde:

$Y_i$  — resultado da soma de  $n$  entradas de um neurônio  $j'$ ;

$x_j$  — estado de ativação do neurônio  $j$ ;  
 $w_{ij}$  — peso da conexão entre  $i$  e  $j$ ;

O Perceptron aceita em sua entrada valores binários e contínuos. Esta rede representou um grande avanço nas aplicações de RNAs, mas existem muitos problemas que não podem ser resolvidos utilizando Perceptron de única camada.

O Perceptron é construído por um neurônio não-linear. O nó deste modelo neural calcula uma combinação linear de entradas juntamente com seus pesos e também é introduzido um limiar externo. É aplicado junto à soma resultante o limitador abrupto, onde o neurônio produz uma saída +1 se a entrada do limitador abrupto for positiva e -1 se a entrada for negativa.[HAY 01]

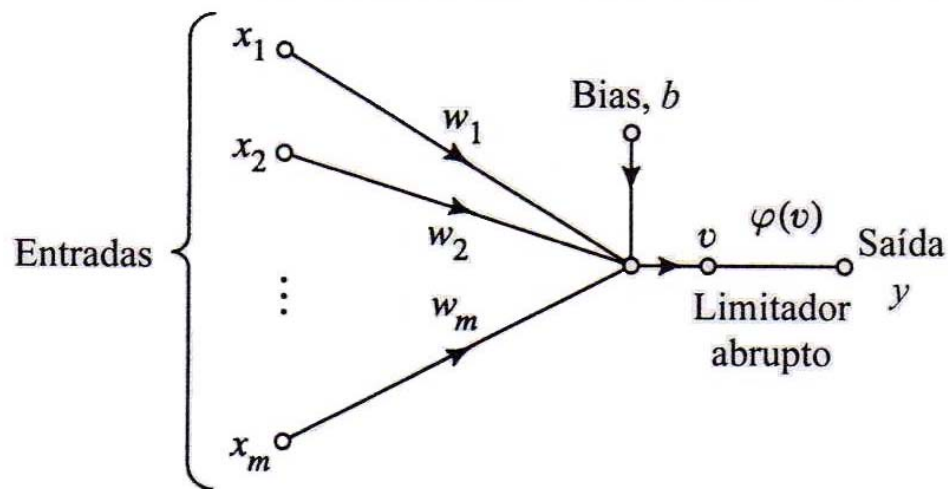


Figura 3.7 Fluxo do Sinal do Perceptron

Fonte: Kovács

A figura acima é o fluxo de sinal do Perceptron, onde  $x$  representa as entradas,  $w$  os pesos sinápticos e  $b$  o bias. A entrada do limitador abrupto é dada pela seguinte equação:

$$v = \sum_{i=1}^m w_i x_i + b \quad (3.7)$$

O objetivo do Perceptron é classificar os estímulos de entrada em duas classes **C1** e **C2**. A regra de decisão funciona da seguinte forma: se a saída  $y$  do Perceptron for igual a  $+1$  a entrada é classificada com **C1**, mas caso seja igual a  $-1$ , a entrada será classificada com **C2**. [HAY 01]

Para visualizar o comportamento de um classificador de padrões, é utilizado um mapa das regiões de decisão no espaço  $m$ -dimensional, abrangido pelas  $m$ -entradas. No Perceptron mais simples existem apenas duas regiões separadas por um hiperplano definido por [HAY 01]

$$\sum_{i=1}^m w_i x_i + b = 0 \quad (3.8)$$

A figura abaixo representa o hiperplano das duas classes de decisão do Perceptron.

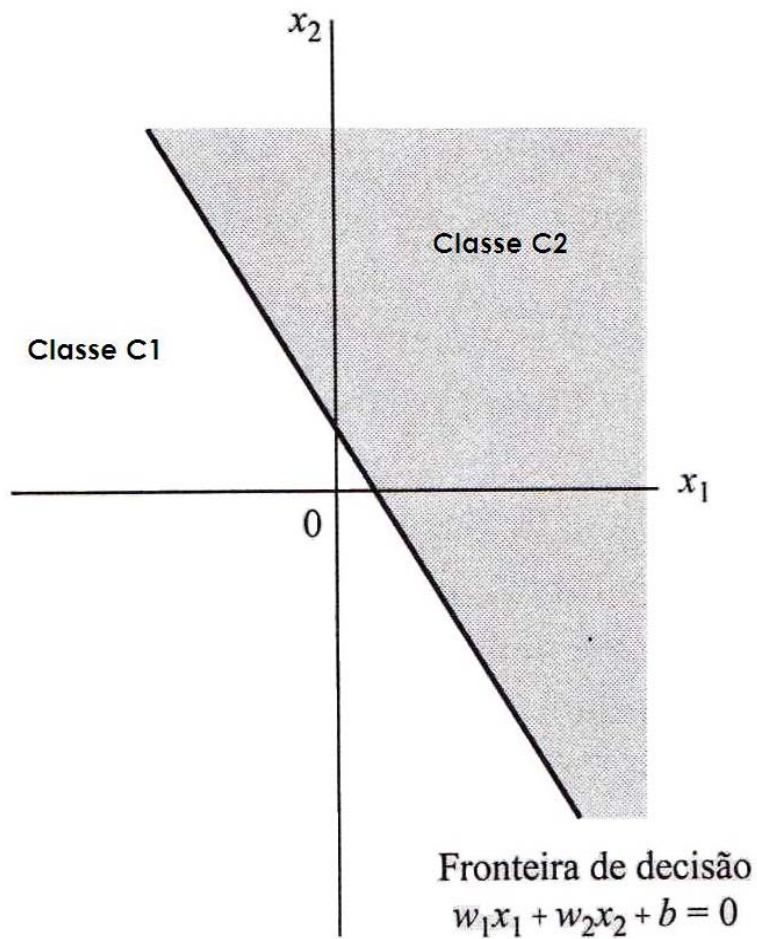


Figura 3.8 Hiperplano com fronteira de Decisão

Fonte: Kovács

O Perceptron utiliza, para treinamento, a seguinte expressão, que é chamada de *Lei de Aprendizado do Perceptron*, proposto por Rosenblatt:

$$\Delta w_i = \eta h(\delta - y_i^d w^t x_i^d) y_i^d x_{i,l}^d \quad (3.9)$$

Onde:

- $\eta$  — taxa de aprendizado
- $h(u)$  — função degrau unitário

### 3.5 Adaline

Enquanto Rosenblatt desenvolvia o Perceptron, Widrow desenvolveu um modelo neural linear, que é muito simples conceitualmente, chamado de ADALINE (acrônimo de ADaptive LINear Element). Este modelo é semelhante ao Perceptron, ele utiliza o algoritmo de aprendizado *LMS (Least Mean Square)*, também conhecido como Regra Delta, onde mais tarde seria generalizado para redes mais elaboradas. [KOV 96]

A diferença entre o Adaline e o Perceptron é a base para as leis de aprendizado. O Adaline aprende usando uma lei de redução dos erros LMS, para reduzir os erros entre a saída atual e a desejada, enquanto o Perceptron utiliza a diferença ponderada simples.

O Adaline é um elemento linear, com isso a saída  $y$  é sempre em função de uma combinação linear do vetor de entrada  $x$ :

$$y = w^t x \quad (3.10)$$

Vale lembrar que este tipo de rede neural não será utilizada neste projeto, devido o Adaline ser utilizado em problemas lineares, que não é objeto de estudo.

### 3.6 MLP

O Perceptron também pode conter camadas ocultas, conhecidas como Perceptron multi-camada (*Multi-layer Perceptron – MLP*). A rede apresenta, além da camada de entrada e de saída, uma ou mais camadas intermediárias.

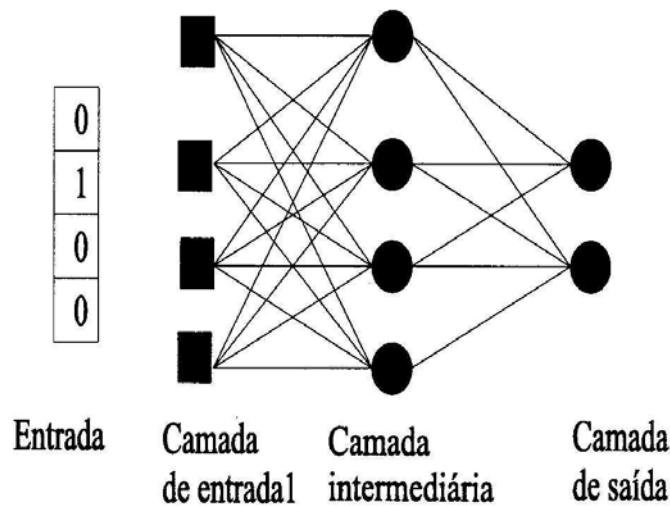


Figura 3.9 Rede MLP com uma camada oculta

Fonte: Braga

Nesse tipo de rede cada camada exerce uma função específica. A camada de saída recebe os estímulos da camada intermediária e constrói o padrão que será a resposta. As camadas intermediárias funcionam como extratora de características, seus pesos são uma codificação de características apresentadas nos padrões de entrada e permite que a rede crie sua própria representação do problema.

### 3.6.1 Back-propagation

O back-propagation é o treinamento que mais se destaca na área de RNA de múltiplas camadas. Ele é atualmente o mais utilizado e o mais eficiente para reconhecimento de padrões. Esse algoritmo de treinamento foi fundamental para a revitalização das redes neurais.

O back-propagation é um treinamento do tipo supervisionado, ou seja, utiliza pares de entrada e de saída desejadas para a correção de erros. O back-propagation ocorre nos dois sentidos da rede, forward (para frente) e backward (para trás). O forward é quem gera a saída, ele faz o sentido normal da rede, indo da entrada até a



saída. Já o backward faz o sentido inverso, ele utiliza a saída desejada e a saída fornecida para ajustar os pesos de cada conexão.[BCL 00]

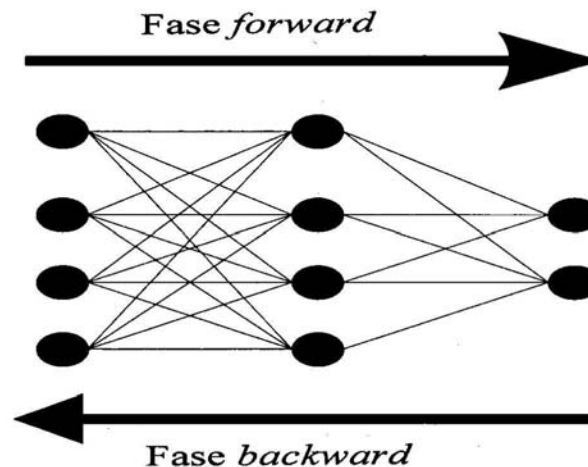


Figura 3.10 Fluxo do processamento do algoritmo back-propagation

Fonte: Kovács

Cada fase do processamento do back-propagation segue um passo distinto, e que foi descrito pelo [BCL 00 pág. 60-61] da seguinte forma:

A fase forward:

1. A entrada é apresentada à primeira camada da rede, a camada  $C^0$ .
2. Para a camada  $C^i$  a partir da camada de entrada
  - 2.1 Após os nodos da camada  $C^i$  ( $i > 0$ ) calcularem seus sinais de saída, estes servem como entrada para a definição das saídas produzidas pelos nodos da camada  $C^{i+1}$ .
3. As saídas produzidas pelos nodos da última camada serão comparadas as saídas desejadas.

A fase backward:

1. A partir da última camada, até a camada de entrada:
  - 1.1 Os nodos da camada atual ajustam seus pesos de forma a reduzir seus erros.

1.2 O erro de um nodo de cada camada intermediária é calculado utilizando os erros dos nodos da camada seguintes conectados a ele, ponderados pelos pesos das conexões entre eles.

O algoritmo back-propagation como um todo:

1. Inicializar pesos e parâmetros.
2. Repetir até o erro ser mínimo ou até a realização de um dado número de ciclos:
  - 2.1 Para cada padrão de treinamento X.
    - 2.1.1 Definir saídas de rede através da fase forward.
    - 2.1.2 Comparar saídas produzidas com as saídas desejadas.
    - 2.1.3 Atualizar pesos dos nodos através do backward.

O algoritmo back-propagation é conhecido também como regra delta generalizada, pois se baseia na regra delta. Este algoritmo define uma forma de descobrir o erro das camadas intermediárias, podendo assim ajustar os pesos da melhor forma possível. Estes pesos serão ajustados com a utilização do gradiente.

O erro total cometido pela rede é dado pela equação abaixo:

$$E = \frac{1}{2} \sum_p \sum_{i=1}^k (d_i^p - y_i^p)^2 \quad (3.11)$$

Onde:

- $E$  — é a medida de erro total  
 $p$  — é o número de padrões  
 $k$  — é o número de unidades de saída  
 $d_i$  — é a  $i$ -ésima saída desejada  
 $y_i$  — é a  $i$ -ésima saída gerada

A fórmula generalizada da correção do erro é dada pela equação abaixo:

$$\Delta w_{ji} = \eta \delta_j x_i \quad (3.12)$$

Ou:

$$w_{ji}(t+1) = w_{ji}(t) + \eta \delta_j(t) x_i(t) \quad (3.13)$$

Onde:

- $w_{ji}$  — o peso das conexões entre a entrada  $x_i$  e o nodo  $j$ .
- $\eta$  — número de conexões de entrada do nodo  $j$ .
- $\delta_j$  — erro no nodo  $j$ .
- $x_i$  — entrada  $i$ .

## Capítulo 4 – Protótipo

### 4.1 Ferramenta Utilizada

Para o desenvolvimento do projeto de reconhecimento de vogais foi utilizado o software Borland C++, no início foi adotada a linguagem C++, porém esta não estava totalmente habilitada em relação aos requisitos necessários. Esta linguagem, que é orientado a objeto, não desempenha as tarefas com a mesma velocidade que a linguagem C “pura” pode oferecer. Este é apenas um dos motivos em que este projeto não foi desenvolvido em um visual mais didático.

Para a realização dos testes foi utilizado um laptop com processador Intel® Pentium® M de 1,6 GHz, 512MB de memória RAM e 100 GB de disco rígido.

A realização deste projeto não visa o interesse comercial, tendo a intenção apenas para fins acadêmicos e de pesquisa.

### 4.2 Métodos e Recursos Utilizados

Foram utilizados no desenvolvimento deste projeto os conceitos de redes neurais artificiais do tipo Perceptron sem camada oculta e com uma camada oculta.

### 4.3 Distribuição do Desenvolvimento do Projeto

Inicialmente obteve-se uma idéia de criação de vários módulos, que individualmente desempenhariam apenas uma função dentro de todo o escopo. E fazendo uma analogia em relação ao nosso sistema Solar foram dados os nomes de planetas e planetóides a cada módulo. Esse procedimento foi feito na ordem inversa de proximidade em relação ao Sol.

São eles:

- Sedna

- Plutão
- Netuno
- Urano
- Saturno

Serão utilizados ao todo, entre treinamento e reconhecimento, 2000 imagens em formato *bitmap* na dimensão de 100x100 pixels. Sendo que estas imagens são divididas igualmente entre as vogais maiúsculas e minúsculas, ou seja, existem 200 imagens de cada vogal maiúscula e 200 de cada minúscula ('A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u').

#### 4.3.1 Disposição dos Arquivos

Os programas Sedna e Plutão foram criados com o intuito de fazer um estudo em relação ao ordenamento dos arquivos das imagens a serem dispostas na rede, ou seja, cada um desses programas gera uma seqüência diferente dos nomes dos arquivos das imagens a serem treinadas e reconhecidas.

O programa Sedna tem como função distribuir os arquivos a serem treinados e reconhecidos. As 2000 imagens a serem utilizadas devem ser divididas entre treinamento e reconhecimento em 3 diferentes agrupamentos (3x1, 2x2, 1x3), por exemplo, 3x1 significa que 3 arquivos de imagens devem ser treinados e apenas 1 deve ser reconhecido. Porém, dentre estes agrupamentos existem várias formas de posicionar os arquivos de imagem, por exemplo, os 3 primeiros serão treinados e o 4º reconhecido. Vale lembrar que sempre serão divididos em grupos de 4 arquivos. Serão 8 diferentes posicionamentos de 3x1, 10 diferentes de 2x2 e 8 diferentes de 1x3, sendo ao todo 26 arquivos.

Abaixo segue um quadro com um exemplo de agrupamento 3x1 com o posicionamento descrito logo acima, mostrando a distribuição entre arquivo de treinamento e arquivo de reconhecimento.

Tabela 4.1 Amostra da distribuição gerada pelo Sedna.

Arquivo de treinamento com o nome das imagens a serem treinadas	Arquivo de reconhecimento com o nome das imagens a serem reconhecidas
Vog0_000 Vog0_001 Vog0_002	Vog0_003
Vog0_004 Vog0_005 Vog0_006	Vog0_007
...	...
Vog9_196 Vog9_197 Vog9_198	Vog9_199

Como pode ser observado no quadro acima, o arquivo com o nome das imagens foi apresentado da seguinte forma: 'Vog0\_000', onde 'Vog' é a abreviação de vogal, junto à abreviação de 'Vog' vem um número:

Tabela 4.2 Representação das vogais

Número	Representa
0 (zero)	vogal 'A' (maiúscula)
1 (um)	vogal 'a' (minúscula)
2 (dois)	vogal 'E' (maiúscula)
3 (três)	vogal 'e' (minúscula)
4 (quatro)	vogal 'I' (maiúscula)
5 (cinco)	vogal 'i' (minúscula)
6 (seis)	vogal 'O' (maiúscula)
7 (sete)	vogal 'o' (minúscula)
8 (oito)	vogal 'U' (maiúscula)
9 (nove)	vogal 'u' (minúscula)

Depois do *under-line* ‘\_’ vem um seqüencial que pode variar de 000 a 199, representando as 200 imagens que serão utilizadas.

O programa Sedna grava primeiramente todas as imagens da vogal ‘A’ (maiúscula), depois todas as imagens da vogal ‘a’ (minúscula) e assim até a vogal ‘u’ (minúscula) que é a última a ser gravada.

O programa Plutão é feito de forma parecida em relação Sedna, e o que difere um do outro é que o Plutão grava em ordem do seqüencial sendo primeiro gravado o seqüencial 000 de todas as vogais, depois o 001 e assim por diante. No Sedna, a ordem de gravação é em relação a vogal. No exemplo que usamos no Sedna, que é o agrupamento 3x1, também será utilizado para o Plutão. Quando se fala em 3x1, no Plutão pode-se entender 30x10, ou seja, grava-se três arquivos das dez vogais maiúsculas e minúsculas, totalizando 30 para treinamento e depois se grava um arquivo de cada, totalizando 10 para reconhecimento. Em vez de serem agrupados de 4 em 4, como no Sedna, agrupa-se de 40 em 40, porém pode ser chamado de 3x1 como antes. Segue abaixo a demonstração do exemplo.

Tabela 4.3 Amostra da distribuição gerada pelo Plutão.

Arquivo de treinamento com o nome das imagens a serem treinadas	Arquivo de reconhecimento com o nome das imagens a serem reconhecidas
Vog0_000 Vog1_000 Vog2_000 Vog3_000 Vog4_000 Vog5_000 Vog6_000 Vog7_000 Vog8_000 Vog9_000 Vog0_001 Vog1_001 Vog2_001 Vog3_001 Vog4_001 Vog5_001 Vog6_001 Vog7_001 Vog8_001 Vog9_001 Vog0_002 Vog1_002 Vog2_002 Vog3_002 Vog4_002 Vog5_002 Vog6_002 Vog7_002 Vog8_002 Vog9_002	Vog0_003 Vog1_003 Vog2_003 Vog3_003 Vog4_003 Vog5_003 Vog6_003 Vog7_003 Vog8_003 Vog9_003
Vog0_004 Vog1_004 Vog2_004 Vog3_004 Vog4_004 Vog5_004 Vog6_004 Vog7_004 Vog8_004 Vog9_004 Vog0_005 Vog1_005 Vog2_005 Vog3_005 Vog4_005 Vog5_005 Vog6_005 Vog7_005 Vog8_005 Vog9_005 Vog0_006 Vog1_006 Vog2_006 Vog3_006 Vog4_006 Vog5_006 Vog6_006 Vog7_006 Vog8_006 Vog9_006	Vog0_007 Vog1_007 Vog2_007 Vog3_007 Vog4_007 Vog5_007 Vog6_007 Vog7_007 Vog8_007 Vog9_007
...	...
Vog0_196 Vog1_196 Vog2_196 Vog3_196 Vog4_196 Vog5_196 Vog6_196 Vog7_196 Vog8_196 Vog9_196 Vog0_197 Vog1_197 Vog2_197 Vog3_197 Vog4_197 Vog5_197 Vog6_197 Vog7_197 Vog8_197 Vog9_197 Vog0_198 Vog1_198 Vog2_198 Vog3_198 Vog4_198 Vog5_198 Vog6_198 Vog7_198 Vog8_198 Vog9_198	Vog0_199 Vog1_199 Vog2_199 Vog3_199 Vog4_199 Vog5_199 Vog6_199 Vog7_199 Vog8_199 Vog9_199

Estes dois programas acima descritos foram criados com a intenção de verificar qual o efeito que a simples mudança da ordem dos arquivos para serem treinados e reconhecidos pode influenciar no desempenho da RNA. Com isso a intenção é averiguar qual seqüência gera o menor erro, com qual seqüência uma vogal tem um melhor ou pior aproveitamento entre outras análises.

### 4.3.2 Vetorização da Imagem

Criado para transformar as imagens *bitmap* em vetores de números reais, o programa Netuno tem como função vetorizar a imagem. Este programa é um dos mais importantes do conjunto, pois os vetores de números reais criados por ele serão objetos de entrada da rede neural artificial.

Para o Netuno poder fazer a vetorização por linha, que são os vetores de números reais, ele leva em consideração alguns dados que são passados pelo usuário, tais como:

- Número de Níveis
- Número de Pixels no eixo X
- Número de Pixels no eixo Y

De um modo geral a vetorização por linha é simplesmente a decomposição da matriz que foi gerada pelo programa Netuno em linhas que serão objetos de entrada da rede neural.

### 4.3.3 Geração dos Pesos

O módulo responsável para a geração dos pesos sinápticos é o Urano. Este módulo irá criar os pesos na quantidade necessária em relação ao número de camadas que terá a rede utilizada e o número de neurônios que cada camada possui.



Este módulo irá gerar os pesos aleatórios que variam de  $-0.5$  a  $+0.5$ . Pode-se sugerir também que os pesos comecem sempre zerados ou então variando entre  $-1$  e  $+1$ , porém estas demais variações não serão objeto de estudo deste projeto. Com os pesos gerados para cada nó da rede neural, todos os pré-requisitos necessários para a execução da rede neural artificial já foram descritos.

#### 4.3.4 Rede Neural Artificial

O programa Saturno é responsável pelo desenvolvimento da rede neural artificial. Neste módulo é apresentada a rede neural Perceptron de camada simples, não existindo camadas ocultas.

Ele recebe um arquivo de texto que contém entre outras informações a quantidade de vezes que a rede irá executar para cada disposição de imagens que está sendo testada. Dentro deste arquivo texto existe na primeira linha o número do contador, ou seja, quantas disposições diferentes das imagens serão treinadas e reconhecidas. Nas demais linhas do arquivo texto contêm o nome do arquivo de treinamento, seguido do nome do arquivo de reconhecimento, o nome do arquivo que contem os pesos sinápticos, um nome de arquivo de extensão '.html' que serão gravadas as informações que foram obtidas nessa rede, o fator de correção dos pesos sinápticos e por último é apresentada a quantidade de vezes que a rede será executada para os arquivos informados anteriormente.

O arquivo de extensão '.html' será de grande valia para a análise dos testes que será descrita no próximo capítulo. O fator de correção do erro dos pesos sinápticos é utilizado quando for comparada a saída desejada com a saída obtida, multiplica-se este fator de correção para tentar minimizar o erro da rede.

Existe também uma variação do programa Saturno, em que será implementada a rede neural Back-propagation. Esta rede receberá os mesmos parâmetros que a rede Perceptron recebeu, diferenciando apenas o arquivo de peso, que contem a quantidade de camadas irá ter a rede, atribuindo os pesos sinápticos para cada camada informada.

Na execução é claro que a rede Back-propagation terá sua particularidade em relação à rede Perceptron.

## Capítulo 5 - Simulações e Resultados

Este capítulo foi dividido em duas partes, a primeira parte abordará a simulação do projeto, ou seja, irá mostrar as etapas que são necessárias para a realização completa do projeto. Na segunda parte, serão apresentados os resultados obtidos em todo o projeto.

Na primeira parte, serão abordadas todas as fases descritas no capítulo anterior, tais como: disposição dos arquivos das imagens; vetorização das imagens; geração dos pesos e a execução da rede neural artificial.

Na figura abaixo, está representado o fluxo dos programas que serão simulados posteriormente.

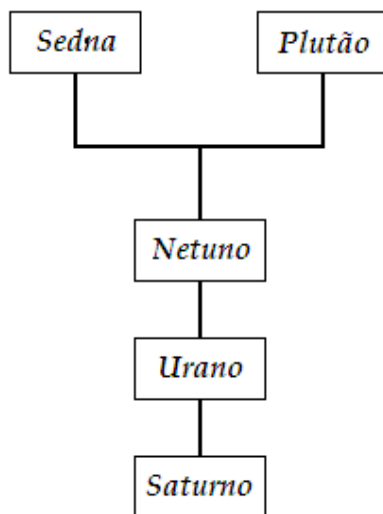


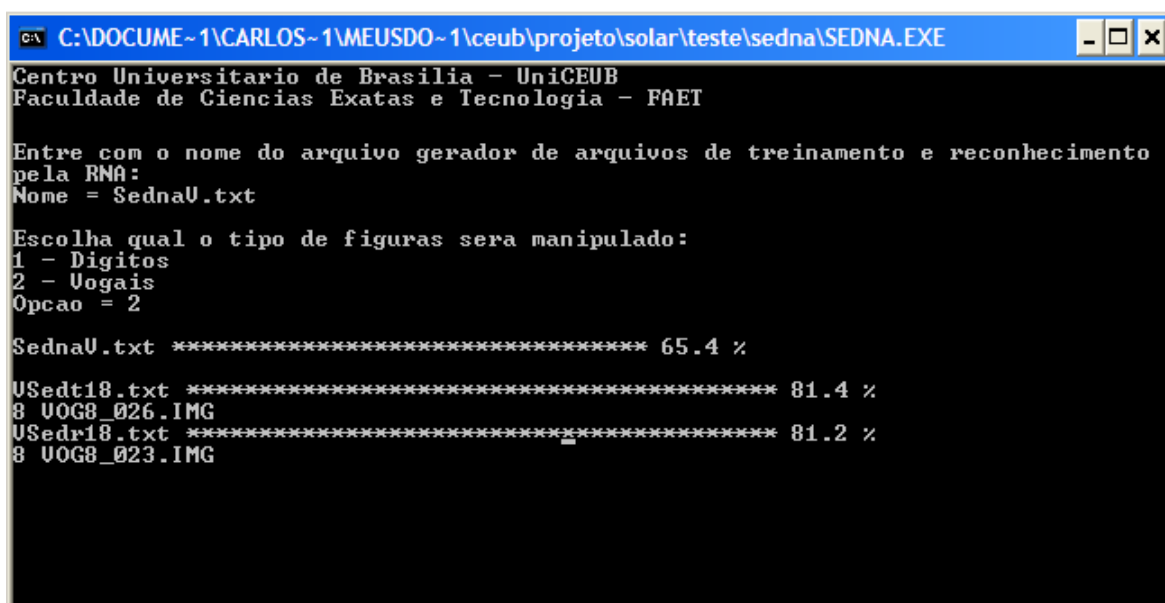
Figura 5.1 Fluxo dos programas.

Os programas foram desenvolvidos com o auxílio e orientação do professor e orientador Aderlon Queiroz, pois ele tem interesse em utilizar estes programas para realizar uma pesquisa mais avançada. Por esse motivo os programas terão algumas funções a mais, porém não serão objetos de estudo desse projeto.

## 5.1 Execução do Programas

### 5.1.1 Disposição da Ordem das Imagens

Este módulo irá fornecer as seqüências de imagens para que possam ser treinadas e reconhecidas posteriormente. Abaixo segue uma imagem da execução do programa Sedna.



```
C:\DOCUMENTOS\CARLOS~1\MEUSDO~1\ceub\projeto\solar\teste\sedna\SEDNA.EXE
Centro Universitario de Brasilia - UniCEUB
Faculdade de Ciencias Exatas e Tecnologia - FAET

Entre com o nome do arquivo gerador de arquivos de treinamento e reconhecimento
pela RNA:
Nome = SednaU.txt

Escolha qual o tipo de figuras sera manipulado:
1 - Digitos
2 - Vogais
Opcao = 2

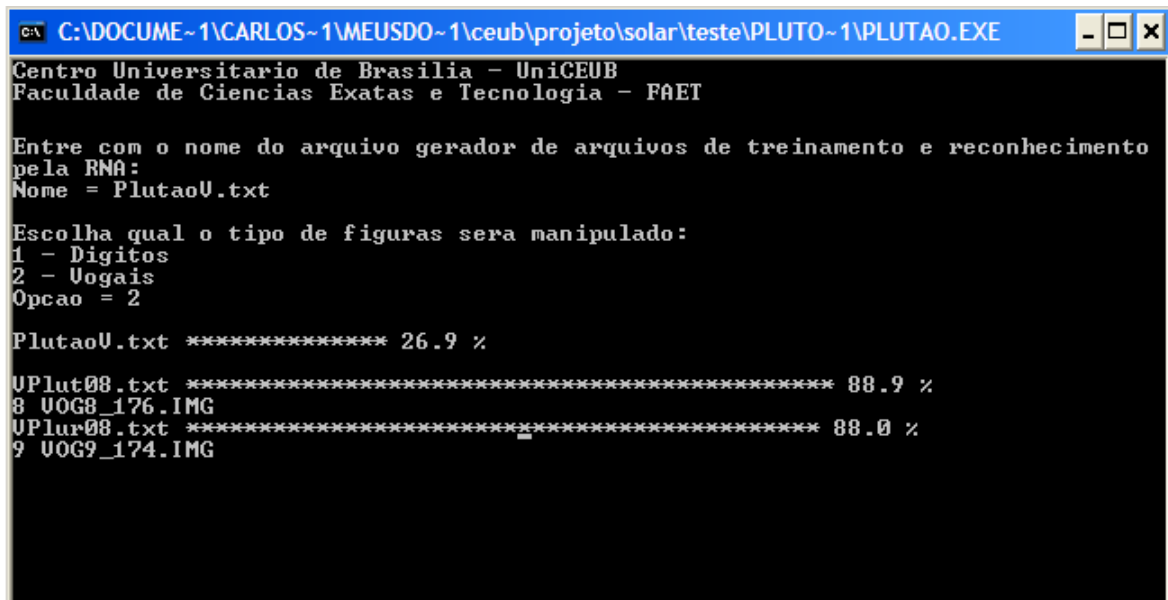
SednaU.txt ***** 65.4 %
VSedt18.txt ***** 81.4 %
8 UOG8_026.IMG
VSedr18.txt ***** 81.2 %
8 UOG8_023.IMG
```

Figura 5.2 Execução do programa Sedna.

No programa Sedna, como já foi mencionado anteriormente, é gerada uma série de arquivos texto. Na figura anterior, primeiramente é solicitado ao usuário que informe o nome do arquivo que será utilizado para gerar os arquivos de treinamento e de reconhecimento, depois é necessário que o usuário escolha qual o tipo de figura que será manipulada, porém apenas as vogais serão abordadas neste projeto. Com as informações recebidas pelo programa serão gerados dois nomes de arquivos: um com os arquivos das imagens que serão treinadas e outro com os arquivos que serão reconhecidos. A figura mostra o exato momento da criação dos arquivos 'VSedt18.txt' e 'VSedr18.txt' que contêm dentro deles os nomes das imagens de treinamento e

reconhecimento, respectivamente. Neste programa serão criados ao todo 26 arquivos diferentes de treinamento e 26 de reconhecimento.

O outro programa que gera uma seqüência de imagens que devem ser processadas é o Plutão. A diferença entre os dois programas já foi descrita no decorrer deste trabalho. Abaixo segue a imagem da execução do programa Plutão.



```
C:\DOCUME~1\CARLOS~1\MEUSDO~1\ceub\projeto\solar\teste\PLUTO~1\PLUTAO.EXE
Centro Universitario de Brasília - UniCEUB
Faculdade de Ciencias Exatas e Tecnologia - FAET

Entre com o nome do arquivo gerador de arquivos de treinamento e reconhecimento
pela RNA:
Nome = PlutaoU.txt

Escolha qual o tipo de figuras sera manipulado:
1 - Digitos
2 - Vogais
Opcao = 2

PlutaoU.txt ***** 26.9 %

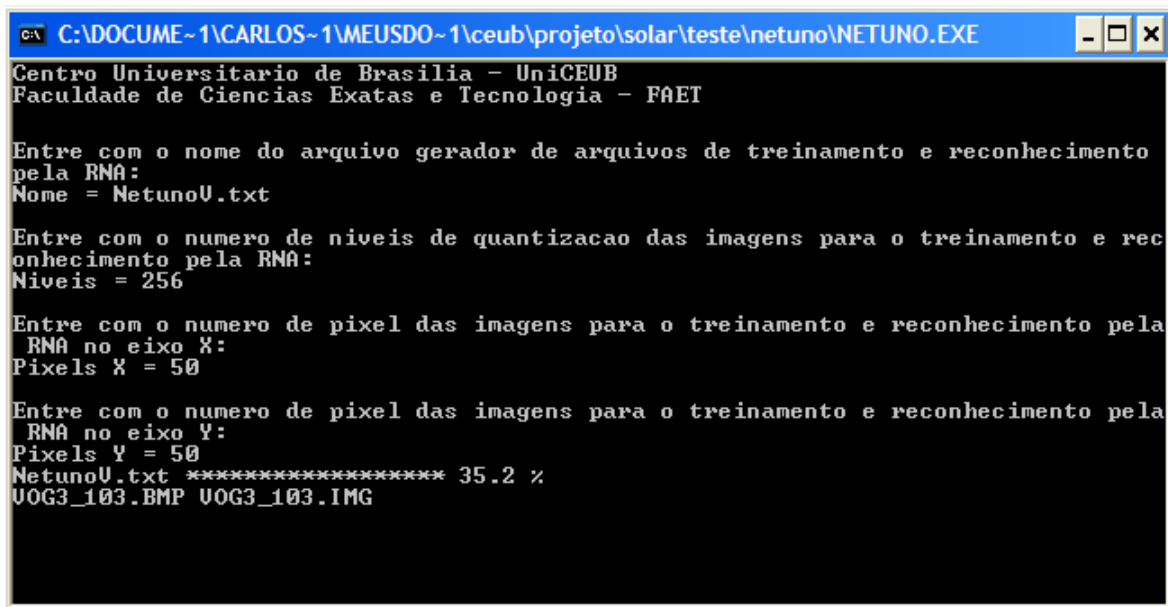
UPlut08.txt ***** 88.9 %
8 UOG8_176.IMG
UPlur08.txt ***** 88.0 %
9 UOG9_174.IMG
```

Figura 5.3 Execução do programa Plutão.

Este programa funciona de forma parecida com o Sedna, ou seja, ele solicita as mesmas informações, porém a forma de gravar a seqüência de arquivo das imagens que é diferente.

### 5.1.2 Vetorização das Imagens

O programa Netuno tem como objetivo vetorizar linearmente as imagens para que estas possam ser utilizadas pela rede neural. Caso não fossem vetorizadas, as imagens não poderiam ser processadas pela rede. O programa é executado da seguinte forma:



```
C:\DOCUME~1\CARLOS~1\MEUSDO~1\ceub\projeto\solar\teste\netuno\NETUNO.EXE
Centro Universitario de Brasilia - UniCEUB
Faculdade de Ciencias Exatas e Tecnologia - FAET

Entre com o nome do arquivo gerador de arquivos de treinamento e reconhecimento
pela RNA:
Nome = NetunoU.txt

Entre com o numero de niveis de quantizacao das imagens para o treinamento e rec
onhecimento pela RNA:
Niveis = 256

Entre com o numero de pixel das imagens para o treinamento e reconhecimento pela
RNA no eixo X:
Pixels X = 50

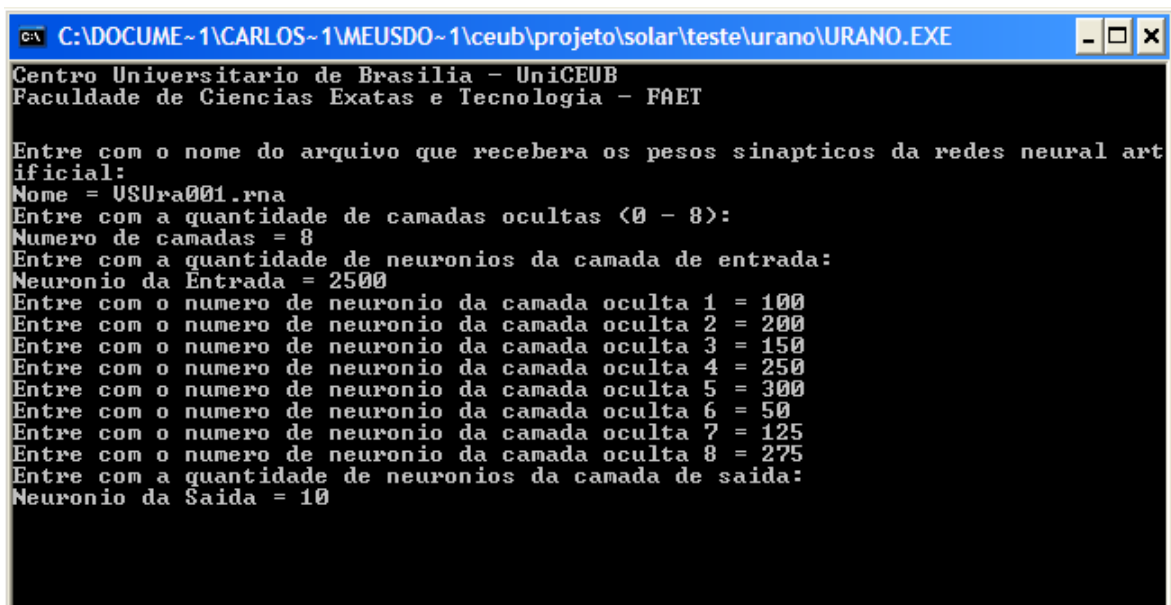
Entre com o numero de pixel das imagens para o treinamento e reconhecimento pela
RNA no eixo Y:
Pixels Y = 50
NetunoU.txt ***** 35.2 %
UOG3_103.BMP UOG3_103.IMG
```

Figura 5.4 Execução do programa Netuno.

Neste programa, é solicitado ao usuário que informe primeiramente qual o arquivo que contém o nome de todas as imagens que serão vetorizadas juntamente com os nomes dos arquivos que irão conter os vetores lineares. Depois, é necessário que o usuário informe o nível de quantização (neste projeto será utilizado apenas o nível 256). Este nível representa a quantidade de tons de cinza que a imagem irá conter. E por fim, o usuário informa, respectivamente, a quantidade de pixels para o eixo x e para o eixo y que a imagem deve ter. No caso, será utilizado 50x50, para que a imagem não fique muita pesada e nem com a qualidade ruim. Com estas informações, o programa gera os vetores com as descrições requisitadas.

### 5.1.3 Geração dos Pesos

Para a geração dos pesos sinápticos será executado o programa Urano. Este programa é o último a ser executado antes da execução do programa que contém a rede neural. Segue a execução da geração dos pesos sinápticos.



```
C:\DOCUME~1\CARLOS~1\MEUSDO~1\ceub\projeto\solar\teste\urano\URANO.EXE
Centro Universitario de Brasilia - UniCEUB
Faculdade de Ciencias Exatas e Tecnologia - FAET

Entre com o nome do arquivo que recebera os pesos sinapticos da redes neural art
ificial:
Nome = USUra001.rna
Entre com a quantidade de canadas ocultas (0 - 8):
Numero de camadas = 8
Entre com a quantidade de neuronios da camada de entrada:
Neuronio da Entrada = 2500
Entre com o numero de neuronio da camada oculta 1 = 100
Entre com o numero de neuronio da camada oculta 2 = 200
Entre com o numero de neuronio da camada oculta 3 = 150
Entre com o numero de neuronio da camada oculta 4 = 250
Entre com o numero de neuronio da camada oculta 5 = 300
Entre com o numero de neuronio da camada oculta 6 = 50
Entre com o numero de neuronio da camada oculta 7 = 125
Entre com o numero de neuronio da camada oculta 8 = 275
Entre com a quantidade de neuronios da camada de saida:
Neuronio da Saida = 10
```

Figura 5.5 Execução do programa Urano.

No programa Urano é solicitado ao usuário que informe qual o nome do arquivo que conterá os pesos sinápticos que serão gerados por este programa. Depois, será necessário que o usuário insira a quantidade de camadas ocultas. No programa desenvolvido é permitido no máximo 8 camadas ocultas, mas em uma rede de multicamadas não existe um número limite. Depois de decidir quantas camadas terá a rede, lembrando que sempre serão 2 camadas ( sendo uma camada de entrada e uma camada de saída ) além do número de camadas ocultas informadas, o usuário digita a quantidade de neurônios que cada camada terá.

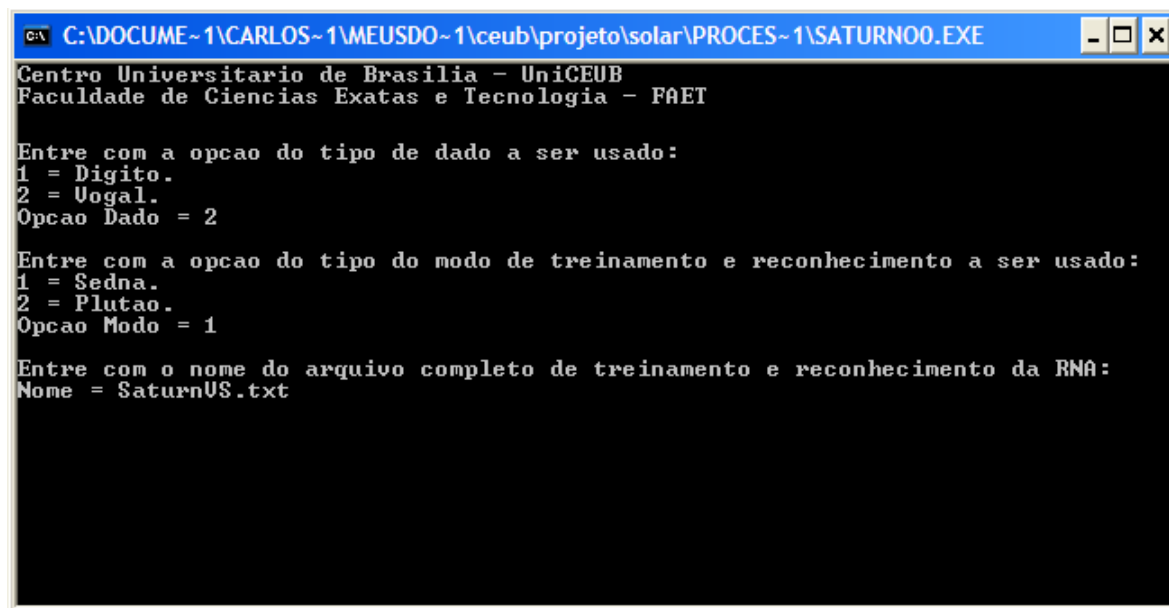
Para este programa, o número de neurônios da camada de entrada deve ser sempre um valor relacionado ao tamanho em que a imagem foi definida no programa Netuno. Por exemplo, se foi informada que a imagem seria de 50x50 serão necessários 2500 neurônios, e se fosse 100x100 então o número de neurônios passaria para 10000, isso ocorre devido à vetorização por linha que é feita no programa Netuno. Nas camadas ocultas, pode se informar qualquer número, lembrando que quanto maior o número de camadas e o número de neurônios, mais lento será o processamento da rede neural. E por último, deve ser informado o número de neurônios da camada de

saída, que para esse projeto será sempre 10, devido à existência de 10 possibilidades de resultados diferentes ('A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u').

Neste projeto, deve conter pesos sinápticos para a rede Perceptron, que não há camadas ocultas, e para a rede Back-propagation, que necessita de um número de camadas ocultas maior que zero.

#### 5.1.4 Rede Neural Artificial

Para desenvolver a rede neural artificial, foi elaborado o programa Saturno que terá duas versões: uma para a rede neural Perceptron e outra para a rede neural Back-propagation. Estas duas versões do Saturno encerram os programas para reconhecimento das vogais manuscritas. Segue abaixo a execução da primeira versão do programa Saturno:



```
C:\DOCUME~1\CARLOS~1\MEUSDO~1\ceub\projeto\solar\PROCES~1\SATURN00.EXE
Centro Universitario de Brasilia - UnICEUB
Faculdade de Ciencias Exatas e Tecnologia - FAET

Entre com a opcao do tipo de dado a ser usado:
1 = Digito.
2 = Vogal.
Opcao Dado = 2

Entre com a opcao do tipo do modo de treinamento e reconhecimento a ser usado:
1 = Sedna.
2 = Plutao.
Opcao Modo = 1

Entre com o nome do arquivo completo de treinamento e reconhecimento da RNA:
Nome = SaturnUS.txt
```

Figura 5.6 Execução da primeira parte do programa Saturno0.

Nesta primeira versão do programa Saturno, que iremos chamar de Saturno0, conterá a rede neural artificial Perceptron. Como entrada de dados do programa teremos primeiramente a informação do tipo de dado, onde continuaremos a informar

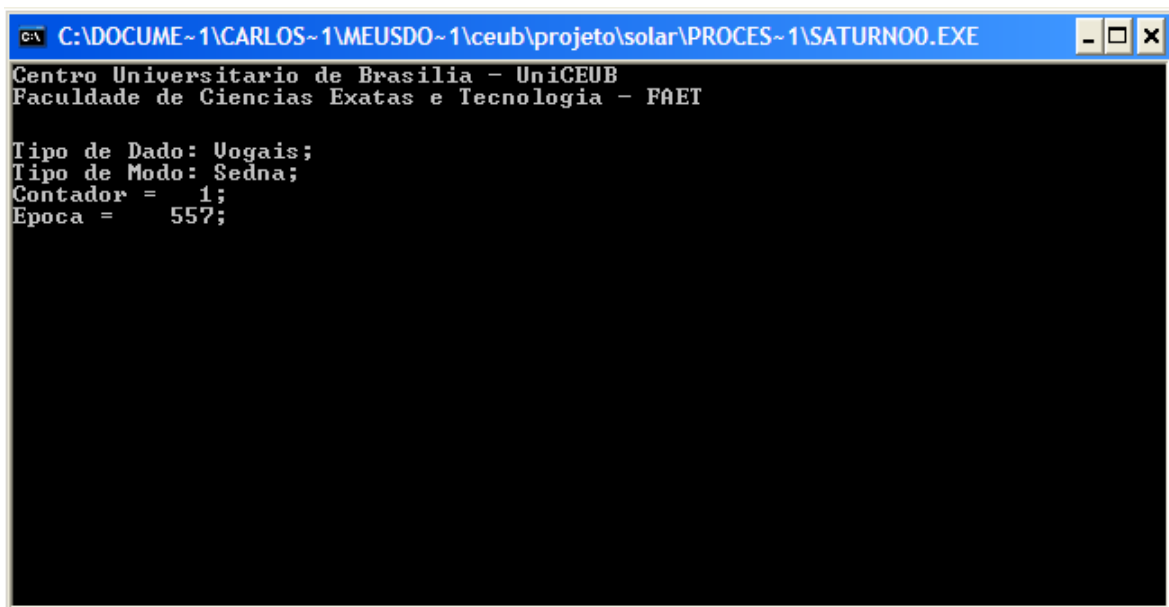


somente a vogal. Depois será informado o tipo de treinamento e reconhecimento, este dado será apenas informativo para ser criado o arquivo 'html' com o tipo de treinamento e reconhecimento que está sendo utilizado. E por último, o usuário informará o nome do arquivo texto onde constará o número de arquivos que serão submetidos à rede neural. Constarão também os nomes de arquivos que serão utilizados para o processamento da rede. O quadro abaixo mostra quais os nomes dos arquivos que constam neste arquivo texto que foi informado pelo usuário.

Tabela 5.1 Amostra da lista que será processada.

VSedt01.txt	VSedr01.txt	VSUra001.rna	VSUra001.html	0.3	5000
VSedt02.txt	VSedr02.txt	VSUra002.rna	VSUra002.html	0.3	5000
VSedt03.txt	VSedr03.txt	VSUra003.rna	VSUra003.html	0.3	5000
VSedt04.txt	VSedr04.txt	VSUra004.rna	VSUra004.html	0.3	5000

A primeira coluna mostra o nome do arquivo que possui o nome das imagens que serão submetidas ao treinamento. A segunda coluna mostra o nome do arquivo que contém o nome das imagens que serão submetidas ao reconhecimento. A terceira coluna mostra o nome do arquivo que compreende os pesos além das características da rede, como, por exemplo, o número de neurônios de cada camada. Na quarta coluna contém nome do arquivo onde serão gravados os resultados obtidos pela execução da rede neural. A quinta coluna possui o número do  $\eta$ , ou melhor, o fator de correção dos pesos sinápticos. E na sexta e última coluna contém o número de interações que a rede será submetida para cada arquivo informado.



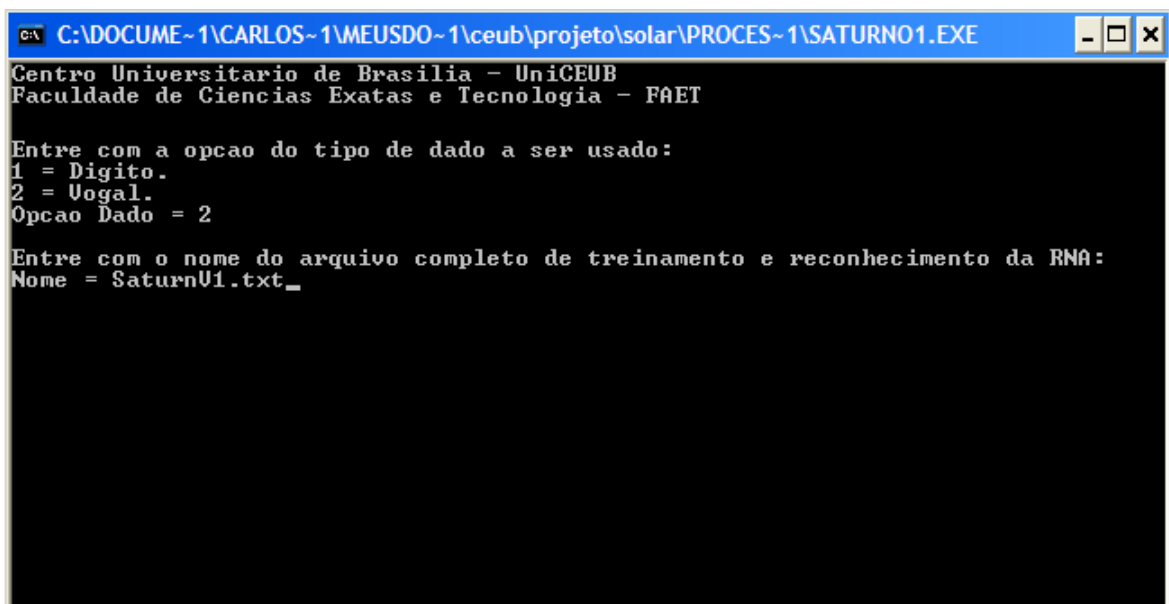
```
C:\DOCUME~1\CARLOS~1\MEUSDO~1\ceub\projeto\solar\PROCES~1\SATURN00.EXE
Centro Universitario de Brasilia - UnICEUB
Faculdade de Ciencias Exatas e Tecnologia - FAET

Tipo de Dado: Uogais;
Tipo de Modo: Sedna;
Contador = 1;
Epoca = 557;
```

Figura 5.7 Continuação da execução do programa Saturno0.

Na figura anterior, podem ser visualizadas as palavras 'Contador' e 'Época' que representam, respectivamente, o número de arquivos a ser processado pela rede neural e o número de interações a ser feito por cada arquivo especificado. No nosso exemplo, o 'Contador' só irá variar somente quando a 'Época' chegar a 5000.

Na segunda versão do programa Saturno, que será chamada de Saturno1, conterá a rede Back-propagation. Essa versão será executada de forma parecida ao Saturno0, mudando apenas alguns detalhes como mostra a figura abaixo.



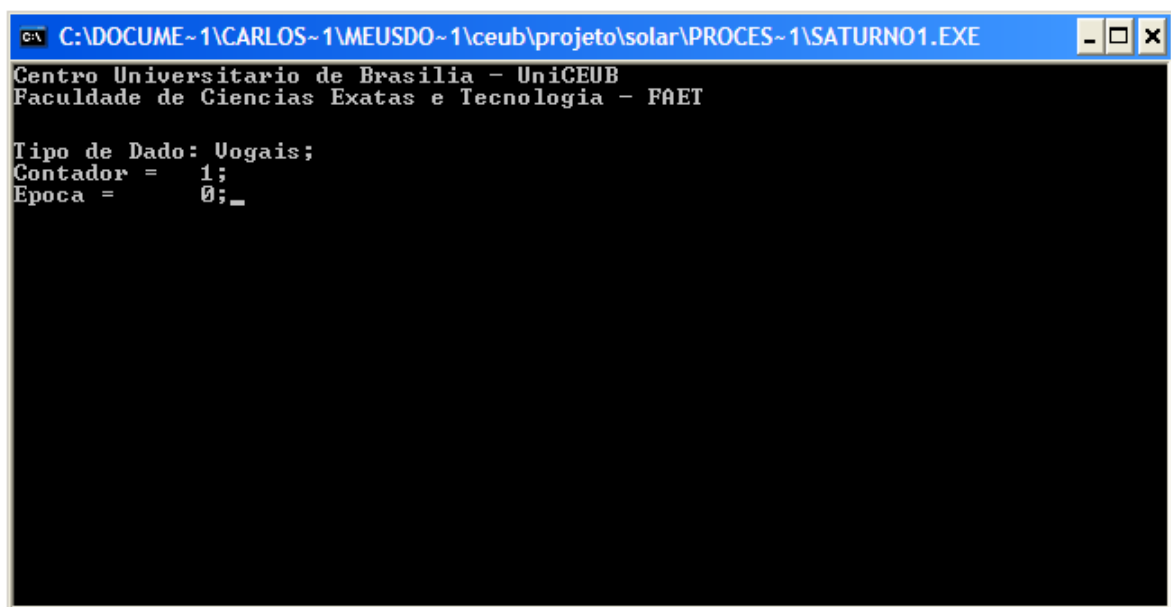
```
C:\DOCUME~1\CARLOS~1\MEUSDO~1\ceub\projeto\solar\PROCES~1\SATURN01.EXE
Centro Universitario de Brasilia - UniCEUB
Faculdade de Ciências Exatas e Tecnologia - FAET

Entre com a opcao do tipo de dado a ser usado:
1 = Digito.
2 = Vogal.
Opcao Dado = 2

Entre com o nome do arquivo completo de treinamento e reconhecimento da RNA:
Nome = SaturnU1.txt_
```

Figura 5.8 Execução da primeira tela do programa Saturno1.

Como se pode perceber, é solicitado ao usuário apenas o tipo de dado e o nome do arquivo que contém as informações a serem utilizadas no processamento da rede neural. Diferente da primeira versão, Saturno0, não é solicitado ao usuário qual o tipo de treinamento e reconhecimento que a rede utilizará. Isso ocorre devido a não separação das imagens entre treinamento e reconhecimento, ou seja, todas as imagens serão treinadas e todas serão reconhecidas.



```
C:\DOCUME~1\CARLOS~1\MEUSDO~1\ceub\projeto\solar\PROCES~1\SATURNO1.EXE
Centro Universitario de Brasilia - UniCEUB
Faculdade de Ciências Exatas e Tecnologia - FAET

Tipo de Dado: Vogais;
Contador = 1;
Epoca = 0;_
```

Figura 5.9 Continuação da execução do programa Saturno1.

Nesta parte da execução, assim como no Saturno0, o 'Contador' irá variar de 1 até o número que foi definido dentro do arquivo texto que foi postado pelo usuário. No caso da 'Época', as ocorrências também irão variar de 1 até o que foi definido, porém a figura mostra a 'Época' em 0 (zero), isso significa que é o início da execução e ainda não foi processado o primeiro arquivo da rede.

## 5.2 Resultados

Nessa segunda parte, serão apresentados os resultados obtidos pelo processamento das redes neurais Perceptron e Back-propagation. Lembrando que a rede Perceptron será representada pelo programa Saturno0 e a rede Back-propagation será representada pelo programa Saturno1.

Nesses dois programas, foi necessária a utilização de grandes áreas de memórias devido ao armazenamento da vetorização das imagens e dos pesos sinápticos de cada neurônio existente. Esses dados devem ser armazenados em ponteiros (tipo de variável da linguagem C) com alocação dinâmica, ou seja, o tamanho do ponteiro varia de acordo com os dados que serão armazenados. Devido a isso, foi



Com o problema apresentado no Saturno1, todos os testes serão baseados na execução do programa Saturno0, mostrando como a disposição das imagens pode influenciar o resultado final. Vale lembrar que será apresentada apenas uma síntese dos resultados, sendo que todos os resultados encontrados estarão disponíveis nos anexos.

No início dos testes foi verificado que o tempo gasto para executar cada arquivo era muito alto. Devido a isso se passou a processar 20 arquivos em vez dos 26, como era a intenção inicial, passando assim a verificar 8 arquivos 3x1 (para cada 3 treinadas, 1 é reconhecida), 8 arquivos 2x2 (para cada 2 treinadas, 2 são reconhecidas) e 4 arquivos 1x3 (para cada imagem treinada, 3 são reconhecidas).

Outra divisão que terá os resultados será em relação ao fator de correção dos pesos sinápticos,  $\eta$ . Serão testados com 2 valores de  $\eta$ , 0,3 e 0,45. Foram escolhidos estes dois valores devido a grande maioria das literaturas pesquisadas para o desenvolvimento deste projeto utilizarem estes dois como referência.

Como exemplo de comportamento da rede neural serão apresentadas algumas partes do arquivo 'html' para mostrar a evolução do reconhecimento.

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	1	17	9	0	0	10	5	1	0	7
a	2	11	6	0	0	21	0	0	0	10
E	4	13	21	0	0	3	1	4	1	3
e	12	12	8	0	0	4	1	0	2	11
I	3	8	22	0	0	4	4	3	2	4
i	0	9	16	0	0	7	6	0	2	10
O	8	24	4	0	1	7	0	2	1	3
o	3	25	3	0	0	6	0	0	2	11
U	4	20	10	0	0	5	1	0	2	8
u	1	21	6	0	0	16	0	0	0	6

Figura 5.11 Início da evolução do reconhecimento.

A figura 5.10 mostra o início do reconhecimento das vogais, mostrando o resultado obtido da primeira interação. Como já foi discutido no capítulo 2, quanto mais vezes a rede é processada, maior é seu aprendizado e com isso ela irá reconhecer mais com o passar das interações. Por isso, pode se notar que a rede ainda está muito longe do que se espera, ou seja, ela ainda não está treinada para o reconhecimento das vogais. Seus pesos sinápticos ainda não estão ajustados corretamente. Neste caso, a rede está identificando 53% de todas as letras como sendo 'a' (minúsculo) e 'E' (maiúsculo). Vale lembrar que todos os testes aqui apresentados terão o número de imagens de cada letra iguais. Para esse caso específico, estão sendo reconhecidas 500 imagens, sendo 50 de cada vogal maiúscula e 50 para cada vogal minúscula.

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	36	5	2	0	1	2	1	1	1	1
a	5	37	1	3	0	0	0	2	1	1
E	2	0	46	0	1	0	1	0	0	0
e	2	3	3	30	1	3	2	4	0	2
I	1	0	1	0	47	0	1	0	0	0
i	1	3	1	1	1	37	0	2	1	3
O	2	3	2	3	0	1	31	3	5	0
o	1	1	1	0	0	2	3	41	1	0
U	3	1	1	0	0	1	5	1	34	4
u	3	4	3	2	0	3	2	1	3	29

Figura 5.12 Metade da evolução do reconhecimento.

A figura 5.11 mostra a metade do processo de evolução do reconhecimento das vogais, ou seja, é apresentada a interação de número 2500 das 5000 que foram feitas. Como já se passaram muitas interações, os pesos sinápticos já estão bem treinados,

apresentando um reconhecimento muito superior ao inicial, passando de um aproveitamento de 9,6% para 73,6%.

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	37	3	3	0	1	2	1	1	1	1
a	5	37	1	3	0	0	0	2	1	1
E	2	0	46	0	1	0	1	0	0	0
e	1	3	4	30	1	3	2	4	0	2
I	1	0	1	0	47	0	1	0	0	0
i	1	3	1	1	1	38	0	1	1	3
O	2	3	2	3	0	1	31	3	5	0
o	1	1	1	0	0	2	3	41	1	0
U	3	1	1	0	0	1	5	1	34	4
u	3	3	3	3	0	2	2	1	3	30

Figura 5.13 Fim da evolução do reconhecimento.

Na figura acima, é apresentado o resultado da última interação da rede neural, sendo o número 5000. Comparando esta interação com a interação de número 2500, não houve muita alteração quanto ao percentual de aproveitamento, aumentando de 73,6% para 74,2%. Isso ocorre devido ao processo de ajuste do erro ser mais perceptível quando há uma ocorrência maior de amostras erradas, porém, a tendência é sempre melhorar o aproveitamento mesmo que discretamente.



### 5.2.1 Sedna utilizando $\eta = 0,3$

#### a) Agrupamento 3x1

Visto o processo de reconhecimento de um modo geral, serão apresentados agora os resultados dos arquivos gerados pelo programa Sedna, utilizando o  $\eta$  igual a 0,3 e o tipo de agrupamento 3x1. Lembrando que os arquivos do programa Sedna são processados na ordem das letras, ou seja, primeiro é processado toda a letra 'A' (maiúscula), depois toda a letra 'a' (minúscula) e assim até processar a letra 'u' (minúscula).

Tabela 5.2 Resultado dos arquivos 3x1 do Sedna, com  $\eta = 0,3$ .

Número do arquivo	A	a	E	e	I	i	O	o	U	u
01	37	37	46	30	47	38	31	41	34	30
02	29	31	42	29	43	37	35	43	31	33
03	37	30	42	35	44	40	38	39	27	34
04	37	33	40	32	45	39	32	41	35	39
15	36	29	42	38	44	35	38	39	38	31
16	34	36	41	29	39	40	38	38	31	29
17	37	35	35	35	46	37	33	36	35	25
18	32	36	39	30	46	37	37	41	35	32

Analisando a tabela acima, pode-se comparar os vários arquivos que para cada 3 arquivos treinados obteve-se um arquivo reconhecido.

Tabela 5.3 Distribuição dos arquivos que utilizam o agrupamento 3x1.

Nome do Arquivo	Distribuição em 3x1 do Sedna (grupos de 4 em 4)	
	Treinamento	Reconhecimento
01	1º, 2º e 3º	4º
02	1º, 2º e 4º	3º
03	1º, 3º e 4º	2º
04	2º, 3º e 4º	1º
15, 16, 17 e 18	Aleatórios	

Pode-se perceber que utilizando esse tipo de agrupamento o melhor padrão é o 'I' (maiúsculo), onde o arquivo 01 reconheceu 47 de 50. Estatisticamente a melhor média foi também para a letra 'I' (maiúscula) com 88,5% de aproveitamento. Porém esta letra não tem o menor desvio padrão (medida mais comum da dispersão estatística), sendo superado pela letra 'i' (minúscula), com desvio de apenas 1,73, e que teve um aproveitamento médio de 75,8%, tendo seu melhor resultado no arquivo 04, onde foram reconhecidos 40 de 50, e o pior aproveitamento no arquivo 15, onde foram reconhecidos 35 de 50.

O padrão 'u' (minúsculo) obteve a pior média do agrupamento 3x1, com 31,63, ou seja, 63.3% de aproveitamento, além do menor valor absoluto encontrado, que foi no arquivo 17 com apenas 25 reconhecidos de 50. A letra 'u' (minúscula) também obteve o maior desvio padrão, com 4,07, variando de 25, no arquivo 17, a 39 no arquivo 04. Com isso é possível afirmar que para esse tipo de agrupamento a letra 'u' é a mais difícil de se reconhecer corretamente.

Dentre os arquivos podemos visualizar que o arquivo 01 obteve o maior valor absoluto com reconhecimento de 47 de 50 para a letra 'I' (maiúscula) e o pior, como já foi relatado, foi a ocorrência de 25 no arquivo 17. A maior média foi obtida no arquivo 04 com 37,3 de 50, obtendo um aproveitamento de 74,6%. E os piores aproveitamentos foram para nos arquivos 02 e 17, com 70,6% e 70,8%. O menor desvio padrão foi obtido pelo arquivo 04, com 4,30 e o maior foi obtido pelo arquivo 01, com 6,15.

Com isso, pode-se concluir que o arquivo 04 é o melhor para esse tipo de disposição, onde a cada quatro imagens, uma é reconhecida. Isso porque este arquivo deixou de reconhecer apenas 127 imagens de 500 que foram processadas. Como este arquivo não tem sua ordem de treinamento gerada de forma aleatória é possível afirmar que: para o agrupamento 3x1, quando a imagem a ser reconhecida é a primeira de um grupo de quatro, se tem uma eficácia superior às demais.

## b) Agrupamento 2x2

Após fazer algumas considerações em relação ao agrupamento 3x1, é a vez do agrupamento 2x2 ser analisado, por isso segue a tabela abaixo com os resultados.

Tabela 5.4 Resultado dos arquivos 2x2 do Sedna, com  $\eta = 0,3$ .

Número do arquivo	A	a	E	e	I	i	O	o	U	u
05	67	67	85	60	90	74	68	76	63	70
06	67	74	82	59	92	74	73	78	64	63
07	64	69	83	62	87	75	71	77	71	66
08	66	55	77	60	84	76	71	77	62	65
09	59	52	77	59	88	78	64	70	69	68
10	74	60	80	70	78	76	69	68	61	70
19	71	60	82	72	90	71	70	79	69	67
20	63	63	74	64	83	74	68	73	76	58

Sendo que os arquivos estão assim divididos:

Tabela 5.5 Distribuição dos arquivos que utilizam o agrupamento 2x2.

Nome do Arquivo	Distribuição em 2x2 do Sedna (grupos de 4 em 4)	
	Treinamento	Reconhecimento
05	1º e 2º	3º e 4º
06	1º e 3º	2º e 4º
07	2º e 3º	1º e 4º
08	1º e 4º	2º e 3º
09	2º e 4º	1º e 3º
10	3º e 4º	1º e 2º
19 e 20	Aleatórios	

Com base nas duas tabelas anteriores, pode-se verificar que novamente a letra 'I' (maiúscula) tem o maior valor absoluto e a maior média em relação às demais letras, esse maior valor absoluto foi 92 de 100, obtido no arquivo 06 e o aproveitamento geral desta letra foi 86,5%.

Com menor número de reconhecimento a letra 'a' (minúscula) obteve uma média geral de 62,5 de no máximo 100, diminuindo seu aproveitamento em relação ao agrupamento 3x1, onde obteve um 66,8%, contra 62,5% de agora. Contrariando essa diminuição de reconhecimento, a letra 'u' (minúscula) melhorou seu aproveitamento, passando de 63,3%, que era a pior do agrupamento 3x1, para 65,9% neste agrupamento. Além da letra 'a' (minúscula) ter a pior média entre as letras, ela obteve também o maior desvio padrão, 7,31, e o menor valor absoluto, sendo 52 de 100.

O menor desvio padrão foi adquirido novamente pela letra 'i' (minúscula), com 2,71, apesar de diminuir a média em relação ao agrupamento 3x1, que foi de 75,8 para 74,8.

A maior diferença entre os valores absolutos de uma letra foi de 22, variando de 52 a 74 que é atribuída a letra 'a' (minúscula) que obteve assim a maior variância dentre as outras letras, com 53,43.

Fazendo uma análise nos resultados dos arquivos, verifica-se que o arquivo 06 obteve o maior valor absoluto com 92 no reconhecimento da letra 'I' (maiúscula). E o menor valor absoluto foi 52, obtido pelo arquivo 09 atribuído ao reconhecimento da letra 'a' (minúscula).

A maior média foi obtida pelo arquivo 06, com 72,6. Porém quem conseguiu o menor desvio padrão foi o arquivo 10, que teve um aproveitamento geral de 70,6%.

O arquivo 09 foi quem obteve a pior média em relação aos demais arquivos, reconhecendo 68,4% das 1000 imagens processadas, e obtendo também o maior desvio padrão, com 10,64.

A partir do resultado do agrupamento 2x2, o arquivo 06, com a ordem de treinar o primeiro arquivo, reconhecer o segundo, treinar o terceiro e reconhecer o quarto, é considerado o melhor arquivo para este tipo de agrupamento.

### c) Agrupamento 1x3

Para terminar com os agrupamentos do Sedna com  $\eta$  igual a 0,3, serão apresentados os resultados do agrupamento 1x3. Neste agrupamento foram testados apenas quatro tipos de arquivos diferentes, pois devido ao tempo de processamento de cada arquivo, foi reduzido de 26 para 20 o número de arquivos processados, sendo assim, os arquivos com agrupamento 1x3 ficaram um pouco prejudicados.

Tabela 5.6 Resultado dos arquivos 1x3 do Sedna, com  $\eta = 0,3$ .

Número do arquivo	A	a	E	e	I	i	O	o	U	u
11	93	95	117	93	128	108	95	102	92	95
12	84	91	128	87	128	108	95	98	99	83
13	98	89	113	93	115	115	101	108	100	92
14	88	71	116	87	122	111	93	103	92	96

Sendo que os arquivos estão assim divididos:

Tabela 5.7 Distribuição dos arquivos que utilizam o agrupamento 1x3.

Nome do Arquivo	Distribuição em 1x3 do Sedna (grupos de 4 em 4)	
	Treinamento	Reconhecimento
11	1º	2º, 3º e 4º
12	2º	1º, 3º e 4º
13	3º	1º, 2º e 4º
14	4º	1º, 2º e 3º

Analisando este último agrupamento verifica-se que novamente a letra 'I' (maiúscula) sobressaiu em relação às demais letras, obtendo a melhor média, com 123,3, ou seja, 82,2% de aproveitamento. Juntamente com a letra 'I' (maiúscula), a letra 'E' (maiúscula) apresentou o maior valor absoluto, 128 de 150 possíveis. Porém, estas duas letras não conseguiram o menor desvio padrão que foi atribuído novamente pela letra 'i' (minúscula), com 3,32, assim como ocorreu com os demais agrupamentos.

A letra 'a' (minúscula) obteve as piores resultados, entre eles estão a menor média com 86,5, sendo apenas 57,7% do total, o maior desvio padrão, com 10,63, e além do menor valor absoluto, que foi de 71 dos 150 possíveis.

Verificando os arquivos pode-se notar que o arquivo 13 obteve a melhor média ponderada com 68,3% do total. Este arquivo também obteve o menor desvio padrão, com 9,8, isso porque não foi este arquivo que obteve o maior valor absoluto, sendo que o arquivo 11 registrou 128 na letra 'I' (maiúscula) e o arquivo 12 registrou 128 nas letras 'I' e 'E' (ambas maiúsculas).

A menor média e o menor valor absoluto foi registrado com o arquivo 14 que obteve 97,9 de média e valor 71 para a letra 'a' (minúscula). O maior desvio padrão foi obtido pelo arquivo 12, com 16,52.

Fazendo uma reflexão ao comportamento da rede neural em relação aos três diferentes agrupamentos, pode-se constatar, e isso já era esperado, que quanto maior é o número de arquivos que são treinados maior é o aproveitamento da rede. No agrupamento 3x1, o aproveitamento geral era de 72,7%, no agrupamento 2x2, era de 71% e no agrupamento 1x3, ficou com 67%. Se formos analisar em relação à letra 'I' (maiúscula) essa diferença fica mais evidente, onde no agrupamento 3x1 era de 88,5%, diminuindo para 86,5% no agrupamento 2x2 e diminuindo ainda mais 4,2% no agrupamento 1x3, chegando a 82,3%.

### 5.2.2 Plutão utilizando $\eta = 0,3$

Terminado a exposição dos resultados do Sedna com  $\eta$  igual a 0,3 será apresentado agora os resultados obtidos com a ordem gerada pelo programa Plutão, lembrando que estes arquivos são divididos de 40 em 40, mas será utilizada a notação atribuída ao Sedna. Será utilizado o mesmo  $\eta$ , 0,3.

### a) Agrupamento 3x1

Para o agrupamento 3x1, é apresentada a tabela abaixo com os resultados obtidos na ordem originada pelo programa Plutão.

Tabela 5.8 Resultado dos arquivos 3x1 do Plutão, com  $\eta = 0,3$ .

Número do arquivo	A	a	E	e	I	i	O	o	U	u
01	38	36	47	28	46	38	36	38	35	26
02	28	30	42	27	43	38	36	43	35	34
03	36	30	43	35	44	42	40	36	35	31
04	35	32	43	34	45	39	34	41	34	37
15	33	31	41	31	43	40	32	43	32	38
16	36	37	41	32	44	34	39	33	36	36
17	34	38	42	36	49	36	39	42	32	30
18	32	30	46	32	46	41	36	40	34	35

Os arquivos seguem a mesma ordem de gravação apresentada na tabela 5.3, mas como já foi relatado, o Plutão gera sua ordem de 40 em 40.

Analisando os resultados obtidos com a execução da rede para a nova ordem de gravação, pode-se verificar que a letra 'I' (maiúscula) obteve uma porcentagem de acerto de 90%. Esta mesma letra também obteve o maior valor absoluto que foi 49 de 50 possíveis. Porém, foi a letra 'U' (maiúscula) que obteve o menor desvio padrão, com 1,46.

A menor média foi atribuída para a letra 'e' (minúscula), onde ficou com 31,88%, sendo 63,8 de aproveitamento. O menor valor absoluto e o maior desvio padrão foi obtido pela letra 'u' (minúscula), reconhecendo apenas 26 dos 50 possíveis no arquivo 01.

Verificando os resultados dos arquivos, pode-se constatar que o arquivo 17 obteve a melhor média, com 75,6% de eficácia, apresentando também a ocorrência do maior valor absoluto, como já foi relatado, na a letra 'I' (maiúscula). O menor desvio padrão foi encontrado no arquivo 16, com 3,68.

O arquivo 02 ficou com a menor média em relação aos demais arquivos de agrupamento, sendo 71,2% de acerto. E o arquivo 01 obteve, além do menor valor absoluto, o maior desvio padrão com 6,6.

Como o melhor aproveitamento entre os arquivos foi obtido pelo arquivo 17 e este tem sua ordem gerada de forma aleatória, não se pode afirmar qual ordem tem melhor aproveitamento para o agrupamento 3x1 gravado de 40 em 40.

## b) agrupamento 2x2

Acabada as análises sobre o agrupamento 3x1 do passa-se para o agrupamento 2x2, com mostra a tabela abaixo.

Tabela 5.9 Resultado dos arquivos 2x2 do Plutão, com  $\eta = 0,3$ .

Número do arquivo	A	a	E	e	I	i	O	o	U	u
05	65	68	87	57	89	75	71	73	65	69
06	69	68	83	60	91	78	75	78	68	64
07	68	68	85	61	89	77	74	80	71	65
08	67	57	77	62	85	79	76	77	61	65
09	59	57	81	60	87	80	70	73	73	69
10	69	60	81	73	77	78	68	70	62	67
19	67	64	68	62	88	73	76	73	68	68
20	59	69	84	70	85	74	76	80	71	67

A ordem de gravação destes arquivos é apresentada na tabela 5.5, mas gravada de 40 em 40.

Verificando o aproveitamento das letras é possível afirmar que a letra 'I' (maiúscula) obteve a melhor média com 86,4% de acerto. A letra 'E' (maiúscula) também obteve uma média alta em relação às demais letras, ficando com 80,8% de aproveitamento. Porém, nem uma destas duas letras obteve o menor desvio padrão, sendo atribuído à letra 'u' (minúscula), com 1,91, vale lembrar que para o grupamento 3x1 esta letra tinha obtido o maior desvio padrão.

O maior valor absoluto ficou novamente com a letra 'I' (maiúscula), obtendo 91 de 100 resultados possíveis.



A menor média foi adquirida pela letra 'e' (minúscula), com aproveitamento de 63,1%, sendo seguida de perto da letra 'a' (minúscula), que obteve uma média de acertos de 63,9 para os 100 que foram processados. O menor valor absoluto foi obtido por estas duas letras minúsculas, 57 de 100 possíveis. O maior desvio padrão foi constatado pela letra 'E' (maiúscula), devido ao baixo reconhecimento encontrado no arquivo 19, que foi de 68 destoando da média que era de 80,8.

Em relação aos arquivos deste agrupamento, o arquivo 07 obteve a maior média geral com 73,8. Porém o maior valor absoluto foi obtido pelo arquivo 06, onde anotou 91 de 100.

Apesar de obter o menor desvio padrão, o arquivo 10 recebeu a média mais baixa entre os outros arquivos deste agrupamento, 70,5. O menor valor absoluto foi obtido pelos arquivos 05, 08 e 09. Sendo que o maior desvio padrão foi adquirido somente pelo arquivo 09.

Para o agrupamento 2x2, gravados de 40 em 40, pode-se afirmar que quando se reconhece o primeiro arquivo de cada letra, depois treina o segundo e o terceiro e, por último, reconhece o quarto, obtém-se um melhor aproveitamento.

### c) Agrupamento 1x3

Serão verificados, na tabela 5.10, os resultados do último agrupamento (1x3) gerado pelo programa Plutão com  $\eta = 0,3$ .

Tabela 5.10 Resultado dos arquivos 1x3 do Plutão, com  $\eta = 0,3$ .

Número do arquivo	A	a	E	e	I	i	O	o	U	u
11	91	92	119	94	126	112	95	99	91	98
12	84	94	130	85	130	110	96	102	96	83
13	99	81	112	91	125	113	102	110	107	92
14	91	69	120	91	124	110	98	105	90	96

Verificando este último agrupamento, percebe-se que a letra 'I' (maiúscula), apesar de diminuir seu aproveitamento, obteve ainda o maior valor com 84,2%. A letra

‘E’ (maiúscula) obteve novamente um aproveitamento elevado, diminuindo apenas 0,6 pontos percentuais, passando de 80,8%, no agrupamento 2x2, para 80,2%. O maior valor absoluto foi obtido por estas duas letras que tiveram as médias mais altas.

A letra ‘i’ (minúscula) obteve o menor desvio padrão com 1,5, tendo uma diferença de apenas 3 reconhecimentos, variando entre 110 e 113. Com esses valores esta letra obteve a 3ª melhor média, com 111,3.

A letra ‘a’ (minúscula) apresentou a menor média, com 84, acertando apenas 56% das imagens. Esta letra foi a que mais perdeu aproveitamento em relação ao agrupamento anterior, que era de 63,9%, perdendo assim 7,9 pontos percentuais.

Analisando os arquivos, verifica-se que o arquivo 13 obteve a melhor média com 103,2, sendo 68,8% de eficácia, porém este arquivo não apresentou o maior valor absoluto, sendo encontrado no arquivo 12.

Por outro lado, a menor média foi atribuída ao arquivo 14, com 99,4, que representa 66,3% de acerto, sendo o único arquivo a não ultrapassar a barreira de 1000 reconhecimentos dos 1500 arquivos analisados.

Fazendo uma análise sobre este agrupamento, o arquivo que reconhece os dois primeiros arquivos de cada letra, depois treina o próximo, e volta a reconhecer o último, obtém o melhor resultado, e esse é o caso do arquivo 13.

Comparando os resultados obtidos com as ordens geradas a partir dos programas Sedna e Plutão com  $\eta$  igual a 0,3, verifica-se que nos dois casos, tanto no Sedna quanto no Plutão, quanto maior o número de imagens a serem treinadas, maiores são os casos positivos de reconhecimento, ou seja, nos agrupamentos 3x1 houveram as maiores taxas de reconhecimentos. Porém entre os arquivos gerado pelo Sedna quem mais se destacou foi o arquivo 04, onde o primeiro é reconhecido e os três posteriores são treinados. E entre os arquivos gerados pelo Plutão, o arquivo que mais se sobressaiu foi o arquivo 17, que teve a sua ordem gerada de forma aleatória. Comparando estes dois arquivos, o arquivo 17, que foi gerado pelo Plutão, obteve o maior aproveitamento com 75,6% contra 75% do arquivo 04 gerado pelo Sedna. Fazendo a média geral de todos os arquivos do Sedna foi de 70,9% de acerto e do Plutão foi de 71,8%, com isso pode-se constatar que a melhor forma em que as

imagens gravadas foram distribuídas, foi a forma que o programa Plutão desenvolveu, ou seja, gravando na ordem do número seqüencial, em vez de gravar na ordem das letras, obtêm-se os melhores aproveitamentos para o reconhecimento para quando o  $\eta$  é 0,3.

Comparando novamente os dois arquivos podemos verificar que sempre a letra 'I' (maiúscula) foi a mais reconhecida, chegando a ser reconhecida 90% das vezes, isso ocorreu no agrupamento 3x1 do Plutão e obtendo um aproveitamento geral no Plutão de 86,8% contra 85,7% no Sedna. Porém, nem todas as letras obtiveram o melhor resultado no Plutão, foi o caso das letras 'A' (maiúscula), 'a' (minúscula) e 'e' (minúscula) que apresentaram perda de 0,8%, 0,3% e 0,2%, respectivamente, quando executados no Plutão. A letra que apresentou a maior diferença percentual entre o Sedna e o Plutão foi a 'O' (maiúscula), que no Sedna obteve a média de acerto de 67,9% passando para 70,5% no Plutão.

Por outro lado, a letra com menor aproveitamento no geral foi a letra 'a' (minúscula) com média de 62,1%.

### 5.2.3 Sedna utilizando $\eta = 0,45$

Serão apresentados os mesmos testes que foram mostrados anteriormente, mudando apenas o valor do  $\eta$ , que passou de 0,3 para 0,45.

#### a) Agrupamento 3x1

Veja a seguir a tabela com os resultados obtidos para os arquivos gerados pelo programa Sedna, com agrupamento 3x1, e  $\eta$  igual a 0,45.

Tabela 5.11 Resultado dos arquivos 3x1 do Sedna, com  $\eta = 0,45$ .

Número do arquivo	A	a	E	e	I	i	O	o	U	u
01	36	35	44	31	47	36	31	37	32	30
02	28	32	43	29	40	37	35	42	33	34
03	37	30	42	35	43	41	37	38	31	33
04	37	32	39	34	45	39	33	41	36	39
15	34	29	40	38	43	35	35	37	34	32
16	37	35	41	30	38	38	40	38	31	31
17	37	35	37	33	46	37	32	34	38	26
18	33	34	37	30	45	37	37	38	32	32

Neste agrupamento é possível verificar que a letra 'I' (maiúscula) obteve a maior média entre as letras, com 47,38 sendo 86,8% de eficácia, obtendo também a ocorrência do maior valor absoluto, 47 de 50 possíveis. Porém o menor desvio padrão foi adquirido pela letra 'i' (minúscula), com 1,85, variando entre 35 e 41.

No lado negativo está a letra 'u' (minúscula) com aproveitamento de 64%, sendo forçado pelo pior valor absoluto, de 26, ocorrendo no arquivo 17. Além disso, esta letra também obteve o maior desvio padrão, que foi de 3,68.

Visualizando os arquivos, percebe-se que o arquivo 04 além de obter o melhor aproveitamento, com 75%, não reconhecendo apenas 125 das 150 imagens que lhe foram dadas, obteve o menor desvio padrão, com 3,95. Porém, o maior valor absoluto foi obtido pelo arquivo 01, com 47 para a letra 'I' (maiúscula).

Sendo o arquivo 04 o melhor para esse tipo de agrupamento e com  $\eta$  igual a 0,45, o arquivo que obteve a menor aproveitamento, com 70,6%, foi o arquivo 02. Porém o maior desvio padrão foi adquirido pelo arquivo 01, 5,67, e o menor valor absoluto, como já foi relatado foi encontrado no arquivo 17, 26 de 50 possíveis.

## b) Agrupamento 2x2

A próxima tabela mostra os resultados obtidos no agrupamento 2x2 com  $\eta$  de 0,45.

Tabela 5.12 Resultado dos arquivos 2x2 do Sedna, com  $\eta = 0,45$ .

Número do arquivo	A	a	E	e	I	i	O	o	U	u
05	67	67	86	57	89	76	68	75	58	68
06	68	65	84	60	91	73	73	80	64	65
07	62	71	84	60	85	77	72	79	72	66
08	64	56	78	60	84	74	73	77	59	67
09	60	53	80	57	89	77	62	72	66	63
10	77	57	80	71	79	77	65	70	59	71
19	68	64	83	72	91	72	69	76	66	66
20	66	61	71	60	81	73	65	73	76	60

Observando os resultados deste agrupamento, é possível verificar que a letra 'I' (maiúscula) obteve a maior média com 86,1, sendo que o maior valor absoluto encontrado foi 91 de 100 possíveis para esta mesma letra. Porém, o menor desvio padrão de 2,1 foi adquirido pela letra 'i' (minúscula), que oscilou apenas entre 72 a 77.

No entanto, a menor média assumida foi da letra 'a' (minúscula), com 61,8, e seguida de perto da média da letra 'e' (minúscula) que obteve uma média de 62,1. Além de assumir a menor média, a letra 'a' (minúscula) obteve também um desvio padrão alto, com 6,11, onde seus reconhecimentos variaram entre 53, que foi o menor valor absoluto entre todas as letras, e 71. O desvio padrão mais alto que o da letra 'a' (minúscula), foi obtido pela letra 'U' (maiúscula), com 6,48.

Verificando os arquivos, pode-se perceber que o arquivo 07 assumiu a maior média, que foi de 72,8. Apesar disso, quem encontrou o maior valor absoluto foi o arquivo 06, com 91 de 100. Em relação ao menor desvio padrão, o arquivo 20 foi quem menos variou, com desvio de 7,29.

Negativamente, o arquivo 09 obteve os piores resultados para esse agrupamento, sendo a menor média com 67,9%, o menor valor absoluto encontrado, que foi 53, além do maior desvio padrão, com 11,34, que variou seus resultados entre 53 e 89.

### c) Agrupamento 1x3

A tabela abaixo apresenta os resultados do agrupamento 1x3.

Tabela 5.13 Resultado dos arquivos 1x3 do Sedna, com  $\eta = 0,45$ .

Número do arquivo	A	a	E	e	I	i	O	o	U	u
11	93	93	117	92	128	109	92	102	89	97
12	85	94	128	82	130	110	96	99	98	86
13	96	86	107	92	113	114	99	105	96	91
14	85	73	118	89	123	110	95	102	91	96

Nesse agrupamento é possível verificar que a letra 'I' (maiúscula) com uma média de 123,5, obteve o melhor aproveitamento. Além disso, esta mesma letra obteve o maior valor absoluto, com 130 de 150 possíveis.

Em relação ao desvio padrão, a letra 'i' (minúscula) obteve o menor com 2,22 e a letra 'a' (minúscula) obteve a maior com 9,68, além do menor aproveitamento, que foi de 57,7%.

A letra 'a' (minúscula) também apresentou o menor valor absoluto com apenas 73, ou seja, no arquivo 14, a taxa de acerto foi inferior a 50%. Impulsionado por este fato, o arquivo 14 obteve um aproveitamento de 65,5%, o menor entre os arquivos deste agrupamento. Porém, quem obteve o maior desvio padrão foi o arquivo 12, com 16,93.

O melhor aproveitamento foi do arquivo 11 com 67,4%, não sendo muito melhor que o do arquivo 14. E o menor desvio padrão, que foi de 9,5, ficou com o arquivo 13.

Terminado a apresentação dos resultados para o Sedna com  $\eta$  igual a 0,45, serão comparados abaixo os dois resultados gerados pela ordem gerada pelo programa Sedna, com  $\eta = 0,3$  e  $\eta = 0,45$ .

Analisando primeiramente o aproveitamento das letras, é possível afirmar que 95% dos casos tem-se melhor aproveitamento quando se é utilizado o  $\eta = 0,3$ , sendo apenas para a letra 'u' (minúscula) o contrário. A maior diferença entre os dois

resultados é no reconhecimento da letra 'o' (minúscula), pois quando se utiliza o  $\eta = 0,3$  obtém-se um aproveitamento de 74,3%, contra 73,2% quando é utilizado  $\eta = 0,45$ .

Em relação aos arquivos, como aconteceu com as letras, a maioria dos arquivos que utilizarem  $\eta = 0,3$  obtiveram os melhores resultados, mas em cerca de 35% isso não ocorreu. A maior diferença foi encontrada no arquivo 15, quando foi utilizado o  $\eta = 0,3$ , o aproveitamento foi de 74,0% e quando foi utilizado  $\eta = 0,45$ , o aproveitamento diminuiu para 71,4%. Mas de um modo geral a diferença foi baixa entre os arquivos, sendo de apenas 0,49%.

#### 5.2.4 Plutão utilizando $\eta = 0,45$

##### a) Agrupamento 3x1

Segue abaixo os resultados dos arquivos gerados pelo programa Plutão e com  $\eta = 0,45$ .

Tabela 5.14 Resultado dos arquivos 3x1 do Plutão, com  $\eta = 0,45$ .

Número do arquivo	A	a	E	e	I	i	O	o	U	u
01	40	36	45	28	47	37	37	38	33	25
02	28	33	42	28	42	36	36	45	34	33
03	36	32	41	33	44	41	40	36	33	31
04	35	29	44	35	44	39	33	41	35	38
15	35	32	40	31	44	40	29	43	32	38
16	34	37	41	33	44	33	38	33	37	27
17	35	36	42	36	49	35	38	42	30	30
18	34	31	47	32	46	41	37	39	33	34

Percebe-se na tabela acima que a letra 'I' (maiúscula) além de obter a maior média, obteve também o maior valor absoluto, tendo um aproveitamento de 90%.

O desvio padrão da letra 'U' (maiúscula) foi considerado o menor, com 2,07. Sendo que o maior foi de 4,21, ocorrido na letra 'u' (minúscula).

Com um aproveitamento de 64%, as letras 'e' e 'u' obtiveram as menores médias com 32. Porém, somente a letra 'u' apresentou o menor valor absoluto neste agrupamento de 3x1, que foi de 25 de 50.

O arquivo 18 obteve o maior aproveitamento com 74,8%, sendo que a média ficou em 37,4. O maior valor absoluto encontrado foi reconhecido pelo arquivo 17, com 49 na letra 'l' (maiúscula). O menor desvio padrão foi apresentado pelo arquivo 04, onde ficou em 4,52.

O arquivo 01 obteve tanto o menor valor absoluto, quanto o menor desvio padrão, que foi de 6,79. No entanto, foram os arquivos 02 e 16 quem obtiveram os menores aproveitamentos, com 71,4% cada um.

#### d) Agrupamento 2x2

A seguir, segue a tabela 5.15 com os resultados do agrupamento 2x2.

Tabela 5.15 Resultado dos arquivos 2x2 do Plutão, com  $\eta = 0,45$ .

Número do arquivo	A	a	E	e	l	i	O	o	U	u
05	65	68	87	57	89	75	71	73	65	69
06	69	68	83	60	91	78	75	78	68	64
07	68	68	85	61	89	77	74	80	71	65
08	67	57	77	62	85	79	76	77	61	65
09	59	57	81	60	87	80	70	73	73	69
10	69	60	81	73	77	78	68	70	62	67
19	67	64	68	62	88	73	76	73	68	68
20	59	69	84	70	85	74	76	80	71	67

Com 86,4%, o aproveitamento da letra 'l' foi considerado o maior para este agrupamento. Esta letra também obteve o maior valor absoluto, 91 de 100 possíveis.

O desvio padrão obtido pela letra 'u' (minúscula) foi de apenas 1,91, sendo considerado o menor para este agrupamento. No entanto, a letra 'E' (maiúscula) obteve o maior desvio padrão, com 5,97.



A letra 'e' (minúscula) obteve a menor média com 63,13, sendo que o menor valor absoluto encontrado foi nas letras 'e' e 'a' (ambas minúsculas), com a ocorrência de 57 de 100 possíveis.

O arquivo 07 obteve nesse agrupamento o melhor aproveitamento, com 73,8%, apesar de não ter encontrado o maior valor absoluto, que foi feito pelo arquivo 06, com 91. O menor desvio padrão foi obtido pelo arquivo 10 com 6,82.

Forçado pelo baixo reconhecimento na letra 'l' (maiúscula), onde a média era de 86,38, e foi reconhecido apenas 77, o arquivo 10 apresentou o menor aproveitamento entre os demais arquivos, com 70,5%.

### e) Agrupamento 1x3

Segue a seguir a tabela 5.16 com os últimos resultados dos arquivos gerados pelo programa Plutão com  $\eta = 0,45$ .

Tabela 5.16 Resultado dos arquivos 1x3 do Plutão, com  $\eta = 0,45$ .

Número do arquivo	A	a	E	e	l	i	O	o	U	u
11	91	92	119	94	126	112	95	99	91	98
12	84	94	130	85	130	110	96	102	96	83
13	99	81	112	91	125	113	102	110	107	92
14	91	69	120	91	124	110	98	105	90	96

Com 126,25, a letra 'l' (maiúscula) obteve a maior média, com um aproveitamento de 84,2%. O maior valor absoluto foi encontrado em duas letras, 'E' e 'l' (ambos maiúsculas), com 130 de 150.

O menor desvio padrão foi obtido pela letra 'i' (minúscula), com 1,5, variando entre 110 e 113 enquanto o maior desvio foi obtido pela letra 'a' (minúscula), com 11,52, que variou entre 69 e 94.

A letra 'a' (minúscula) também adquiriu o menor aproveitamento, que foi de apenas 56%, e o menor valor absoluto, 69 de 150, que foi gerado pelo arquivo 14, sendo que este valor representa apenas 46% de acerto.

Verificando o aproveitamento dos arquivos, verifica-se que o arquivo 13 obteve o melhor percentual de aproveitamento, com 68,8%, sendo que ele se sobressaiu mais em relação aos demais arquivos no reconhecimento da letra 'U' (maiúscula), onde obteve um aproveitamento de 71,3% enquanto os outros arquivos tiveram uma média de aproveitamento de 61,6%. O arquivo 14 obteve o menor aproveitamento, com 66,3%, sendo que o percentual de acerto da letra 'a' foi o menor entre os demais, 46% contra 59,3% de média dos demais.

Comparando os arquivos gerados pelo Plutão com  $\eta = 0,3$  e  $\eta = 0,45$ , pode-se constatar que, assim como ocorreu nos arquivos do Sedna, os arquivos que utilizaram  $\eta = 0,3$ , obtiveram os melhores resultados, porém com uma diferença menor, obtendo um aproveitamento de apenas 0,18% superior em relação aos arquivos que utilizaram  $\eta = 0,45$ .

A maior diferença entre o reconhecimento foi dada na letra 'u' (minúscula), onde foi reconhecido 65% das amostras nos arquivos com  $\eta = 0,3$ , contra 64,1% nos que utilizaram  $\eta = 0,45$ .

Fazendo uma média do desvio padrão, nos dois casos a letra 'i' (minúscula) obteve o menor desvio, com 2,2 para  $\eta = 0,3$  e 2,32 para  $\eta = 0,45$ . E o maior foi encontrado na letra 'a' (minúscula), com 6,7 quando  $\eta = 0,3$  e 6,5 para  $\eta = 0,45$ .

Em relação aos arquivos, o arquivo 17, que utilizou  $\eta = 0,3$  e foi processado no agrupamento 3x1, obteve o melhor aproveitamento, com 75,6%, e pior aproveitamento foi obtido pelo arquivo 12 com 67,3%, tanto com  $\eta = 0,3$ , quanto com  $\eta = 0,45$ , sendo executado no agrupamento 1x3.

O maior desvio padrão foi obtido pelo arquivo 12 com 17,42, tanto com  $\eta = 0,3$ , quanto com  $\eta = 0,45$ , e o menor pelo arquivo 16 com  $\eta = 0,3$ , que foi de 3,68.

### 5.2.5 Arquivos diversos utilizando $\eta = 0,45$

Para terminar, serão apresentados os resultados de testes gerados a partir de oito seqüências diferentes de arquivos, que foram fornecidos pelo orientador Aderlon

Queiroz. Uma particularidade desse teste foi a utilização de todas as 2000 imagens para serem treinadas e as mesmas 2000 para serem reconhecidas.

Os resultados a seguir foram obtidos com a utilização de  $\eta = 0,45$ .

Tabela 5.17 Resultado para 2000 imagens com  $\eta = 0,45$ .

Nome do arquivo	A	a	E	e	I	i	O	o	U	u
Sedna	187	189	192	176	200	192	194	192	191	190
Quaoar	190	190	192	176	200	191	194	192	193	190
Xena	174	178	192	179	198	191	190	196	180	184
Plutão	172	184	195	179	198	191	191	187	188	186
Urano	174	171	194	179	198	192	189	196	189	184
Netuno	174	176	194	179	198	193	189	187	190	183
Saturno	171	183	194	179	198	191	189	196	187	185
Júpiter	167	185	192	178	198	193	192	195	186	184

Cada arquivo contém uma ordem gravada diferente. Os quatro primeiros geram a ordem em grupos de 4 em 4, da mesma forma em que foi feito pelo programa Sedna utilizado nos testes anteriores. E, os quatro últimos são gerados em grupos de 40 em 40, como feito pelo programa Plutão nos teste anteriores.

Verificando os resultados é possível verificar que novamente a letra 'I' (maiúscula) apresentou os melhores desempenhos, tendo um aproveitamento de 99,3%, chegando a 100% de acerto em dois arquivos. Porém o menor desvio padrão foi obtido pela letra 'i' (minúscula), com 0,89.

Do lado negativo, a letra 'A' (maiúscula) obteve os piores resultados, tendo reconhecido apenas 88,1% das amostras, além de manter o desvio padrão mais elevado, com 8,26.

O arquivo de nome Quaoar obteve o melhor aproveitamento, sendo de 95,4%. Este arquivo segue a ordem da letra, primeiro grava toda a letra 'A' (maiúscula), depois a letra 'a' (minúscula) e assim até a letra 'u' (minúscula). Porém a ordem do número seqüencial é aleatória dentro dos grupos de 4 em 4.

Tabela 5.18 Ordem de gravação do arquivo Quaoar.

Arquivo Quaoar										
0 VOG0_003.IMG, 0 VOG0_002.IMG, 0 VOG0_001.IMG, 0 VOG0_000.IMG, 0 VOG0_004.IMG, 0 VOG0_007.IMG, 0 VOG0_006.IMG, 0 VOG0_005.IMG, 0 VOG0_008.IMG, 0 VOG0_010.IMG, 0 VOG0_009.IMG, 0 VOG0_011.IMG, 0 VOG0_014.IMG, 0 VOG0_015.IMG, 0 VOG0_012.IMG, 0 VOG0_013.IMG, 0 VOG0_018.IMG, 0 VOG0_019.IMG, 0 VOG0_016.IMG, 0 VOG0_017.IMG, 0 VOG0_021.IMG, 0 VOG0_022.IMG, 0 VOG0_020.IMG, 0 VOG0_023.IMG ..... 9 VOG9_178.IMG, 9 VOG9_179.IMG, 9 VOG9_177.IMG, 9 VOG9_176.IMG, 9 VOG9_180.IMG, 9 VOG9_182.IMG, 9 VOG9_181.IMG, 9 VOG9_183.IMG, 9 VOG9_186.IMG, 9 VOG9_184.IMG, 9 VOG9_185.IMG, 9 VOG9_187.IMG, 9 VOG9_191.IMG, 9 VOG9_188.IMG, 9 VOG9_189.IMG, 9 VOG9_190.IMG, 9 VOG9_194.IMG, 9 VOG9_192.IMG, 9 VOG9_193.IMG, 9 VOG9_195.IMG, 9 VOG9_199.IMG, 9 VOG9_196.IMG, 9 VOG9_197.IMG, 9 VOG9_198.IMG										

### 5.2.6 Arquivos diversos utilizando $\eta = 0,3$

Depois de apresentado o resultado para  $\eta = 0,45$ , é a vez de ser apresentado o resultado obtido com  $\eta = 0,3$ .

Tabela 5.19 Resultado para 2000 imagens com  $\eta = 0,3$ .

Nome do arquivo	A	a	E	e	I	i	O	o	U	u
Sedna	191	187	191	175	200	192	193	193	190	186
Quaoar	191	187	191	175	200	192	192	191	192	186
Xena	175	177	191	181	198	192	190	197	188	185
Plutão	172	181	194	179	198	192	192	194	187	186
Urano	174	173	194	179	198	192	191	196	188	184
Netuno	174	173	193	171	198	192	185	187	189	184
Saturno	171	185	193	178	198	191	190	196	191	186
Júpiter	172	181	193	178	198	191	192	195	186	185

Para este caso, a letra 'i' obteve os mesmos resultados que foram obtidos quando foi utilizado o  $\eta = 0,45$ , ou seja, um aproveitamento de 99,3%.

O desvio padrão da letra 'i' (minúscula) continuou sendo o menor, porém ele diminuiu neste resultado, foi de 0,89 no teste anterior, para 0,46 neste. O maior desvio padrão também continuou sendo da letra 'A' (maiúscula), porém aumentou de 8,03 para 8,44.

O menor aproveitamento para este teste foi da letra 'e' (minúscula), com 88,5% de acerto.

Analisando os arquivos, percebe-se que o arquivo de nome Sedna obteve o melhor desempenho, com 94,9%, sendo seguido de perto pelo arquivo Quaoar, com 94,85%. O arquivo Sedna é gravado de forma seqüencial, sendo que primeiramente é gravado em ordem das letras.

Tabela 5.20 Ordem de gravação do arquivo Sedna.

Arquivo Sedna
0 VOG0_000.IMG, 0 VOG0_001.IMG, 0 VOG0_002.IMG, 0 VOG0_003.IMG, 0 VOG0_004.IMG, 0 VOG0_005.IMG, 0 VOG0_006.IMG, 0 VOG0_007.IMG, 0 VOG0_008.IMG, 0 VOG0_009.IMG, 0 VOG0_010.IMG, 0 VOG0_011.IMG, 0 VOG0_012.IMG, 0 VOG0_013.IMG, 0 VOG0_014.IMG, 0 VOG0_015.IMG, 0 VOG0_016.IMG, 0 VOG0_017.IMG, 0 VOG0_018.IMG, 0 VOG0_019.IMG, 0 VOG0_020.IMG, 0 VOG0_021.IMG, 0 VOG0_022.IMG, 0 VOG0_023.IMG, ... 9 VOG9_176.IMG, 9 VOG9_177.IMG, 9 VOG9_178.IMG, 9 VOG9_179.IMG, 9 VOG9_180.IMG, 9 VOG9_181.IMG, 9 VOG9_182.IMG, 9 VOG9_183.IMG, 9 VOG9_184.IMG, 9 VOG9_185.IMG, 9 VOG9_186.IMG, 9 VOG9_187.IMG, 9 VOG9_188.IMG, 9 VOG9_189.IMG, 9 VOG9_190.IMG, 9 VOG9_191.IMG, 9 VOG9_192.IMG, 9 VOG9_193.IMG, 9 VOG9_194.IMG, 9 VOG9_195.IMG, 9 VOG9_196.IMG, 9 VOG9_197.IMG, 9 VOG9_198.IMG, 9 VOG9_199.IMG

Comparando agora os dois resultados onde foi feito o reconhecimento das 2000 imagens, pode-se constatar que quando se utiliza  $\eta = 0,45$ , obtêm-se os melhores resultados. Sendo que o desvio padrão mais baixo foi encontrado quando se utilizou  $\eta = 0,3$ , com média de 2,87, contra 3,17 quando utilizado  $\eta = 0,45$ .

O arquivo de nome Quaoar foi o que melhor se apresentou fazendo uma média entre os dois casos.

Agora fazendo uma avaliação geral das letras, percebe-se que, sem dúvida alguma, a letra 'I' (maiúscula) foi a letra que mais obteve acertos em todos os arquivos. Isso se deve ao fato que a letra 'I' (maiúscula) tem a sua escrita mais simples, e que a maioria das pessoas a escreve da mesma forma. Vale também perceber que, na maioria das vezes, as letras maiúsculas são mais reconhecidas que as minúsculas, isso porque normalmente as pessoas escrevem a letra maiúscula da mesma forma que a letra de forma, criando assim um padrão de escrita.

## Capítulo 6 – Conclusão

### 6.1 Considerações Finais

Com o desenvolvimento deste trabalho foi possível reconhecer, como foi apresentado nos resultados obtidos, grande parte das imagens que foram utilizadas como base.

Nesta monografia, foram apresentados os conceitos de reconhecimento de padrão, além da apresentação da Rede Neural Artificial. Isso com a intenção de abordar os temas que foram utilizados no decorrer do projeto.

A utilização de Inteligência Artificial foi interessante por ser uma área em expansão e com grande potencial. As Redes Neurais Artificiais, que estão dentro da Inteligência Artificial, vêm aumentando seus adeptos a cada ano, com isso tem-se investido muito dinheiro em pesquisas nesta área.

Este projeto visou ser uma parte do estudo necessário para a realização de grandes desenvolvimentos, como, por exemplo, a implementação do radar inteligente que envolve uma tecnologia muito avançada.

A pesar do não funcionamento total do módulo que continha a rede neural com a utilização do algoritmo de back-propagation, o resultado obtido no reconhecimento de caracteres foi concluído com um aproveitamento satisfatório.

E, por fim, o projeto foi importante para o crescimento profissional, tendo como possíveis conseqüências uma pós-graduação ou projetos ainda mais complexos, sendo também importante para incentivar esse tipo de projeto dentro da própria instituição de ensino.

## 6.2 Dificuldades Encontradas

Dentre as principais dificuldades encontradas, destacam-se:

- Desenvolvimento do algoritmo de Back-propagation.
- Obtenção dos resultados muito demorada.

No desenvolvimento do algoritmo Back-propagation foi constatado a dificuldade em relação ao armazenamento da imagem e dos pesos, como já foi explicado no decorrer da monografia.

Na obtenção dos resultados foi observada uma grande demora, isso porque para a rede neural obter um bom resultado é necessário que haja um grande número de amostras, além de grande número de treinamentos. Com isso, o tempo gasto para a realização dos testes foi extremamente dispendioso.

## 6.3 Sugestão para Trabalhos Futuros

Utilizando os conceitos apresentados neste trabalho, pode-se ter como inspiração de trabalhos futuros:

- O reconhecimento de dígitos, também manuscritos, com a utilização de redes neurais.
- O reconhecimento de figuras geométricas.
- O reconhecimento de outras letras do alfabeto ocidental.
- Apresentar um estudo mais profundo de redes neurais artificiais.
- Fazer uma interligação entre redes neurais e mineração de dados (DATAMINIG).

## Referências

- [BCL 00] BRAGA, Antônio de Pádua; CARVALHO, André Ponce de L. F.; LUDERMIR, Teresa Bernarda. **Redes Neurais Artificiais Teoria e Aplicações**. LTC, 2000
- [HAY 01] HAYKIN, Simon. **Redes Neurais: princípios e prática**. 2ª Edição – Ed. Bookman. Porto Alegre – RS, 2001.
- [JPJ 06] Jesan, John Peter. **The neural approach to pattern recognition**, 2005. Disponível em: [http://www.acm.org/ubiquity/views/v5i7\\_jesan.html](http://www.acm.org/ubiquity/views/v5i7_jesan.html) - acessado em outubro 2006.
- [KOV 96] KOVÁCS, Zsolt L. **Redes Neurais Artificiais – Fundamentos e Aplicações**. 2ª Edição, Ed. Collegium Cognitio. São Paulo – SP, 1996.
- [LF 99] LIMA FILHO, Adimael Barbosa. **Reconhecimento de Caracteres**. Canoas – RS, 1999.
- [NCE 06] Silva, Eugênio, 2006. Disponível em: <http://www.nce.ufrj.br/conceito/artigos/2006/016p1-3.htm> - acessado em outubro de 2006.



# Anexo A – Algoritimos

## A.1 Sedna

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

```
FILE    *fiArqIn,
        *fiArqOut1,
        *fiArqOut2;
```

```
void vApresentacao(void);
```

```
void vObtem_Nome_Arquivo(char *cNome1);
```

```
void vObtem_Tipo_Arquivo(unsigned short *usOpcao);
```

```
void vAbrir_Arquivo_In(const char *cNome1);
```

```
void vAbrir_Arquivo_Out_1(const char *cNome2);
```

```
void vAbrir_Arquivo_Out_2(const char *cNome3);
```

```
void vLimpa_Linha(unsigned short int usLinha);
```

```
void vGrava_Out_1(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha);
```

```
void vGrava_Out_2(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha);
```

```
void vMostrar_Barra_1(const char *cNome1, const int iCont1, const int iAux1, const unsigned short
usLinha);
```

```
void vMostrar_Barra_2(const char *cNome2, const int iCont2, const int iAux1, const unsigned short
usLinha);
```

```
void vMostrar_Barra_3(const char *cNome3, const int iCont3, const int iAux1, const unsigned short
usLinha);
```

```
void vTipo_0(const int iAux1, const unsigned short usOpcao);
```

```
void vTipo_0_Caso_1(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);
```

```
void vTipo_0_Caso_2(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);
```

```
void vTipo_0_Caso_3(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);
```

```
void vTipo_1(const int iAux1, const int iAux2, const unsigned short usOpcao, const char *cNome2, const
char *cNome3);
```

```
void vTipo_2(const int iAux1, const int iAux2, const int iAux3, const unsigned short usOpcao, const char
*cNome2, const char *cNome3);
```

```
void vTipo_3(const int iAux1, const int iAux2, const int iAux3, const int iAux4, const unsigned short
usOpcao, const char *cNome2, const char *cNome3);
```

```
void main(void) {
```

```
    char    cNome1[81],
            cNome2[81],
            cNome3[81];
```

```
    int      iAux1 = 0,
            iAux2 = 0,
            iAux3 = 0,
            iAux4 = 0,
            iCont1 = 0,
            il = 0,
            iTipo1 = 0,
            iTipo2 = 0;
```

```
    unsigned short usOpcao;
```

```
    randomize();
```

```
    vApresentacao();
```

```
    vObtem_Nome_Arquivo(cNome1);
```

```
    vObtem_Tipo_Arquivo(&usOpcao);
```

```
    vAbrir_Arquivo_In(cNome1);
```

```
    fscanf(fiArqIn, "%d%d", &iCont1, &iAux1);
```

```
    gotoxy(1, 18);
```

```
    printf("%s 0.0%%", cNome1);
```

```
    vLimpa_Linha(20);
```

```
    vLimpa_Linha(21);
```

```
    vLimpa_Linha(22);
```

```
    vLimpa_Linha(23);
```

```
    vTipo_0(iAux1, usOpcao);
```

```
    fclose(fiArqOut1);
```

```
    fclose(fiArqOut2);
```

```
    for(il = 0; il < iCont1; il++) {
        fscanf(fiArqIn, "%d", &iTipo1);
        vLimpa_Linha(20);
        vLimpa_Linha(21);
        vLimpa_Linha(22);
        vLimpa_Linha(23);
        switch(iTipo1) {
            case 0 :
```

```
                fscanf(fiArqIn, "%d%s%s", &iTipo2, cNome2, cNome3);
```

```
                switch(iTipo2) {
```

```
                    case 1 :
```

```
                        vTipo_0_Caso_1(iAux1, usOpcao, cNome2, cNome3);
```

```

        break;
        case 2 :
            vTipo_0_Caso_2(iAux1, usOpcao, cNome2, cNome3);
            break;
        case 3 :
            vTipo_0_Caso_3(iAux1, usOpcao, cNome2, cNome3);
            break;
    };
    break;
    case 1 :
        fscanf(fiArqIn, "%d%s%s", &iAux2, cNome2, cNome3);
        vTipo_1(iAux1, iAux2, usOpcao, cNome2, cNome3);
        break;
    case 2 :
        fscanf(fiArqIn, "%d%d%s%s", &iAux2, &iAux3, cNome2, cNome3);
        vTipo_2(iAux1, iAux2, iAux3, usOpcao, cNome2, cNome3);
        break;
    case 3 :
        fscanf(fiArqIn, "%d%d%d%s%s", &iAux2, &iAux3, &iAux4, cNome2, cNome3);
        vTipo_3(iAux1, iAux2, iAux3, iAux4, usOpcao, cNome2, cNome3);
        break;
    };
    fclose(fiArqOut1);
    fclose(fiArqOut2);
    vMostrar_Barra_1(cNome1, il + 1, iCont1, 18);
};
fclose(fiArqIn);
}

void vApresentacao(void) {
    clrscr();
    printf("Centro Universitario de Brasilia - UniCEUB\n");
    printf("Faculdade de Ciencias Exatas e Tecnologia - FAET\n");
}

void vObtem_Nome_Arquivo(char *cNome1) {
    printf("Entre com o nome do arquivo gerador de arquivos de treinamento e reconhecimento pela RNA:\nNome = ");
    scanf("%s", cNome1);
}

void vObtem_Tipo_Arquivo(unsigned short *usOpcao) {
    printf("\nEscolha qual o tipo de figuras sera manipulado:\n1 - Digitos\n2 - Vogais\nOpcao = ");
    scanf("%u", usOpcao);
    if((*usOpcao != 1) && (*usOpcao != 2))
        *usOpcao = 1;
}

void vAbrir_Arquivo_In(const char *cNome1) {

```

```

        fiArqIn = fopen(cNome1, "rt");
        if(fiArqIn == NULL) {
            printf("\nNão consegui abrir o arquivo: %s", cNome1);
            exit(1);
        };
    }

void vAbrir_Arquivo_Out_1(const char *cNome2) {

    fiArqOut1 = fopen(cNome2,"wt");
    if(fiArqOut1 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s",cNome2);
        exit(1);
    };
}

void vAbrir_Arquivo_Out_2(const char *cNome3) {

    fiArqOut2 = fopen(cNome3,"wt");
    if(fiArqOut2 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s",cNome3);
        exit(1);
    };
}

void vLimpa_Linha(unsigned short int usLinha) {

    unsigned short usl;

    for(ysl = 0; ysl < 80; ysl++) {
        gotoxy(ysl, usLinha);
        printf(" ");
    };
}

void vGrava_Out_1(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha) {

    if(usOpcao == 1) {
        fprintf(fiArqOut1, "%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    } else {
        fprintf(fiArqOut1, "%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    };
}

void vGrava_Out_2(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha) {

    if(usOpcao == 1) {
        fprintf(fiArqOut2, "%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    }
}

```

```

        gotoxy(1, usLinha);
        printf("%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    } else {
        fprintf(fiArqOut2,"%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    };
}

```

```

void vMostrar_Barra_1(const char *cNome1, const int iCont1, const int iAux1, const unsigned short usLinha) {

```

```

    int iM;
    float fPorc;

    gotoxy(1, usLinha);
    printf("%s ", cNome1);
    fPorc = (1.0 * iCont1) / iAux1;
    for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
        gotoxy(strlen(cNome1) + 2 + iM, usLinha);
        printf("**");
    };
    gotoxy(strlen(cNome1) + 2 + iM, usLinha);
    printf(" %.1f %%", 100.0*fPorc);
}

```

```

void vMostrar_Barra_2(const char *cNome2, const int iCont2, const int iAux1, const unsigned short usLinha) {

```

```

    int iM;
    float fPorc;

    gotoxy(1, usLinha);
    printf("%s ", cNome2);
    fPorc = (1.0 * iCont2) / iAux1;
    for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
        gotoxy(strlen(cNome2) + 2 + iM, usLinha);
        printf("**");
    };
    gotoxy(strlen(cNome2) + 2 + iM, usLinha);
    printf(" %.1f %%", 100.0*fPorc);
}

```

```

void vMostrar_Barra_3(const char *cNome3, const int iCont3, const int iAux1, const unsigned short usLinha) {

```

```

    int iM;
    float fPorc;

    gotoxy(1, usLinha);
    printf("%s ", cNome3);
    fPorc = (1.0 * iCont3) / iAux1;
    for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
        gotoxy(strlen(cNome3) + 2 + iM, usLinha);

```

```

        printf("**");
    };
    gotoxy(strlen(cNome3) + 2 + iM, usLinha);
    printf(" %.1f %%", 100.0*fPorc);
}

void vTipo_0(const int iAux1, const unsigned short usOpcao) {

    int    iCont1 = 0,
           iCont2 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4];

    if(usOpcao == 1) {
        vAbrir_Arquivo_Out_1("DSedt00.txt");
        vAbrir_Arquivo_Out_2("DSedr00.txt");
    } else {
        vAbrir_Arquivo_Out_1("VSedt00.txt");
        vAbrir_Arquivo_Out_2("VSedr00.txt");
    };
    fprintf(fiArqOut1, "%d\n", iAux1);
    fprintf(fiArqOut2, "%d\n", iAux1);
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
        iMat[il][3] = -1;
    };
    iCont1 = 0;
    for(il = 0; il < 10; il++)
        for(iJ = 0; iJ < 2; iJ++)
            for(iK = 0; iK < 10; iK++)
                for(iL = 0; iL < 10; iL++) {
                    iMat[iCont1][0] = il;
                    iMat[iCont1][1] = iJ;
                    iMat[iCont1][2] = iK;
                    iMat[iCont1][3] = iL;
                    iCont1++;
                };

    iCont1 = 0;
    iCont2 = 0;
    for(il = 0; il < iAux1; il++) {
        vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
        iCont1++;
        if(usOpcao == 1)
            vMostrar_Barra_2("DSedt00.txt", iCont1, iAux1, 20);
        else
            vMostrar_Barra_2("VSedt00.txt", iCont1, iAux1, 20);
    };
    for(il = 0; il < iAux1; il++) {
        vGrava_Out_2(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 23);
    };
}

```

```

        iCont2++;
        if(usOpcao == 1)
            vMostrar_Barra_3("DSedr00.txt", iCont2, iAux1, 22);
        else
            vMostrar_Barra_3("VSedr00.txt", iCont2, iAux1, 22);
    };
}

```

```

void vTipo_0_Caso_1(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {

```

```

    int      iAux2 = 0,
             iCont1 = 0,
             iCont2 = 0,
             iCont3 = 0,
             il = 0,
             iJ = 0,
             iK = 0,
             iL = 0,
             iMat[2000][4],
             iTemp1 = 0;

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.75 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.25 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
        iMat[il][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iTemp1 = 1;
    iAux2 = random(4)+1;
    for(il = 0; il < 10; il++)
        for(iJ = 0; iJ < 2; iJ++)
            for(iK = 0; iK < 10; iK++)
                for(iL = 0; iL < 10; iL++) {
                    if(iTemp1 != iAux2) {
                        iMat[iCont2 * 3 + iCont1][0] = il;
                        iMat[iCont2 * 3 + iCont1][1] = iJ;
                        iMat[iCont2 * 3 + iCont1][2] = iK;
                        iMat[iCont2 * 3 + iCont1][3] = iL;
                        iCont1++;
                    } else {
                        iMat[iCont2 + (int)(0.75 * iAux1)][0] = il;
                        iMat[iCont2 + (int)(0.75 * iAux1)][1] = iJ;
                        iMat[iCont2 + (int)(0.75 * iAux1)][2] = iK;
                        iMat[iCont2 + (int)(0.75 * iAux1)][3] = iL;
                    };
                    iCont3++;
                }

```

```

        if(iCont3 == 4) {
            iCont1 = 0;
            iCont2++;
            iCont3 = 0;
        };
        if(iTemp1 < 4) iTemp1++;
        else {
            iTemp1 = 1;
            iAux2 = random(4)+1;
        };
    };

    iCont1 = 0;
    iCont2 = 0;
    for(il = 0; il < (int)(0.75 * iAux1); il++) {
        vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.75 * iAux1), 20);
    };
    for(il = 0; il < (int)(0.25 * iAux1); il++) {
        vGrava_Out_2(usOpcao, iMat[il + (int)(0.75 * iAux1)][0], iMat[il + (int)(0.75 * iAux1)][1],
iMat[il + (int)(0.75 * iAux1)][2], iMat[il + (int)(0.75 * iAux1)][3], 23);
        iCont2++;
        vMostrar_Barra_3(cNome3, iCont2, (int)(0.25 * iAux1), 22);
    };
}

void vTipo_0_Caso_2(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {

    int    iAux2 = 0,
           iAux3 = 0,
           iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iCont4 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0;

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.5 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.5 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
        iMat[il][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;

```



```

iCont3 = 0;
iCont4 = 0;
iTemp1 = 1;
iAux2 = random(4)+1;
do
    iAux3 = random(4) + 1;
while(iAux3 == iAux2);
for(il = 0; il < 10; il++)
    for(iJ = 0; iJ < 2; iJ++)
        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                if((iTemp1 != iAux2) && (iTemp1 != iAux3)) {
                    iMat[iCont3 * 2 + iCont1][0] = il;
                    iMat[iCont3 * 2 + iCont1][1] = iJ;
                    iMat[iCont3 * 2 + iCont1][2] = iK;
                    iMat[iCont3 * 2 + iCont1][3] = iL;
                    iCont1++;
                } else {
                    iMat[iCont3 * 2 + iCont2 + (int)(0.5 * iAux1)][0] = il;
                    iMat[iCont3 * 2 + iCont2 + (int)(0.5 * iAux1)][1] = iJ;
                    iMat[iCont3 * 2 + iCont2 + (int)(0.5 * iAux1)][2] = iK;
                    iMat[iCont3 * 2 + iCont2 + (int)(0.5 * iAux1)][3] = iL;
                    iCont2++;
                };
                iCont4++;
                if(iCont4 == 4) {
                    iCont1 = 0;
                    iCont2 = 0;
                    iCont3++;
                    iCont4 = 0;
                };
                if(iTemp1 < 4) iTemp1++;
                else {
                    iTemp1 = 1;
                    iAux2 = random(4) + 1;
                    do
                        iAux3 = random(4) + 1;
                    while(iAux3 == iAux2);
                };
            };
};

iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
    iCont1++;
    vMostrar_Barra_2(cNome2, iCont1, (int)(0.5 * iAux1), 20);
};
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_2(usOpcao, iMat[il + (int)(0.5 * iAux1)][0], iMat[il + (int)(0.5 * iAux1)][1],
iMat[il + (int)(0.5 * iAux1)][2], iMat[il + (int)(0.5 * iAux1)][3], 23);
    iCont2++;
    vMostrar_Barra_3(cNome3, iCont2, (int)(0.5 * iAux1), 22);
};
}

```

```

void vTipo_0_Caso_3(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {

    int      iAux2 = 0,
              iAux3 = 0,
              iAux4 = 0,
              iCont1 = 0,
              iCont2 = 0,
              iCont3 = 0,
              il = 0,
              iJ = 0,
              iK = 0,
              iL = 0,
              iMat[2000][4],
              iTemp1 = 0;

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.25 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.75 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
        iMat[il][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iTemp1 = 1;
    iAux2 = random(4)+1;
    do
        iAux3 = random(4) + 1;
    while(iAux3 == iAux2);
    do
        iAux4 = random(4) + 1;
    while((iAux4 == iAux2) || (iAux4 == iAux3));
    for(il = 0; il < 10; il++)
        for(iJ = 0; iJ < 2; iJ++)
            for(iK = 0; iK < 10; iK++)
                for(iL = 0; iL < 10; iL++) {
                    if((iTemp1 != iAux2) && (iTemp1 != iAux3) && (iTemp1 != iAux4))
                    {
                        iMat[iCont2][0] = il;
                        iMat[iCont2][1] = iJ;
                        iMat[iCont2][2] = iK;
                        iMat[iCont2][3] = iL;
                    } else {
                        iMat[iCont2 * 3 + iCont1 + (int)(0.25 * iAux1)][0] = il;
                        iMat[iCont2 * 3 + iCont1 + (int)(0.25 * iAux1)][1] = iJ;
                        iMat[iCont2 * 3 + iCont1 + (int)(0.25 * iAux1)][2] = iK;
                        iMat[iCont2 * 3 + iCont1 + (int)(0.25 * iAux1)][3] = iL;
                        iCont1++;
                    }
                }
}

```

```

        };
        iCont3++;
        if(iCont3 == 4) {
            iCont1 = 0;
            iCont2++;
            iCont3 = 0;
        };
        if(iTemp1 < 4) iTemp1++;
        else {
            iTemp1 = 1;
            iAux2 = random(4) + 1;
            do
                iAux3 = random(4) + 1;
            while(iAux3 == iAux2);
            do
                iAux4 = random(4) + 1;
            while((iAux4 == iAux2) || (iAux4 == iAux3));
        };
    };

    iCont1 = 0;
    iCont2 = 0;
    for(il = 0; il < (int)(0.25 * iAux1); il++) {
        vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.25 * iAux1), 20);
    };
    for(il = 0; il < (int)(0.75 * iAux1); il++) {
        vGrava_Out_2(usOpcao, iMat[il + (int)(0.25 * iAux1)][0], iMat[il + (int)(0.25 * iAux1)][1],
iMat[il + (int)(0.25 * iAux1)][2], iMat[il + (int)(0.25 * iAux1)][3], 23);
        iCont2++;
        vMostrar_Barra_3(cNome3, iCont2, (int)(0.75 * iAux1), 22);
    };
}

```

```

void vTipo_1(const int iAux1, const int iAux2, const unsigned short usOpcao, const char *cNome2, const
char *cNome3) {

```

```

    int      iCont1 = 0,
            iCont2 = 0,
            iCont3 = 0,
            il = 0,
            iJ = 0,
            iK = 0,
            iL = 0,
            iMat[2000][4],
            iTemp1 = 0;

```

```

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.75 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.25 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
    }

```

```

        iMat[iI][2] = -1;
        iMat[iI][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iTemp1 = 1;
    for(iI = 0; iI < 10; iI++)
        for(iJ = 0; iJ < 2; iJ++)
            for(iK = 0; iK < 10; iK++)
                for(iL = 0; iL < 10; iL++) {
                    if(iTemp1 != iAux2) {
                        iMat[iCont2 * 3 + iCont1][0] = iI;
                        iMat[iCont2 * 3 + iCont1][1] = iJ;
                        iMat[iCont2 * 3 + iCont1][2] = iK;
                        iMat[iCont2 * 3 + iCont1][3] = iL;
                        iCont1++;
                    } else {
                        iMat[iCont2 + (int)(0.75 * iAux1)][0] = iI;
                        iMat[iCont2 + (int)(0.75 * iAux1)][1] = iJ;
                        iMat[iCont2 + (int)(0.75 * iAux1)][2] = iK;
                        iMat[iCont2 + (int)(0.75 * iAux1)][3] = iL;
                    };
                    iCont3++;
                    if(iCont3 == 4) {
                        iCont1 = 0;
                        iCont2++;
                        iCont3 = 0;
                    };
                    if(iTemp1 < 4) iTemp1++;
                    else
                        iTemp1 = 1;
                };
    iCont1 = 0;
    iCont2 = 0;
    for(iI = 0; iI < (int)(0.75 * iAux1); iI++) {
        vGrava_Out_1(usOpcao, iMat[iI][0], iMat[iI][1], iMat[iI][2], iMat[iI][3], 21);
        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.75 * iAux1), 20);
    };
    for(iI = 0; iI < (int)(0.25 * iAux1); iI++) {
        vGrava_Out_2(usOpcao, iMat[iI + (int)(0.75 * iAux1)][0], iMat[iI + (int)(0.75 * iAux1)][1],
iMat[iI + (int)(0.75 * iAux1)][2], iMat[iI + (int)(0.75 * iAux1)][3], 23);
        iCont2++;
        vMostrar_Barra_3(cNome3, iCont2, (int)(0.25 * iAux1), 22);
    };
}

void vTipo_2(const int iAux1, const int iAux2, const int iAux3, const unsigned short usOpcao, const char
*cNome2, const char *cNome3) {

    int    iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,

```

```

        iCont4 = 0,
        il = 0,
        iJ = 0,
        iK = 0,
        iL = 0,
        iMat[2000][4],
        iTemp1 = 0;

vAbrir_Arquivo_Out_1(cNome2);
vAbrir_Arquivo_Out_2(cNome3);
fprintf(fiArqOut1, "%d\n", (int)(0.5 * iAux1));
fprintf(fiArqOut2, "%d\n", (int)(0.5 * iAux1));
for(il = 0; il < iAux1; il++) {
    iMat[il][0] = -1;
    iMat[il][1] = -1;
    iMat[il][2] = -1;
    iMat[il][3] = -1;
};
iCont1 = 0;
iCont2 = 0;
iCont3 = 0;
iCont4 = 0;
iTemp1 = 1;
for(il = 0; il < 10; il++)
    for(iJ = 0; iJ < 2; iJ++)
        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                if((iTemp1 != iAux2) && (iTemp1 != iAux3)) {
                    iMat[iCont3 * 2 + iCont1][0] = il;
                    iMat[iCont3 * 2 + iCont1][1] = iJ;
                    iMat[iCont3 * 2 + iCont1][2] = iK;
                    iMat[iCont3 * 2 + iCont1][3] = iL;
                    iCont1++;
                } else {
                    iMat[iCont3 * 2 + iCont2 + (int)(0.5 * iAux1)][0] = il;
                    iMat[iCont3 * 2 + iCont2 + (int)(0.5 * iAux1)][1] = iJ;
                    iMat[iCont3 * 2 + iCont2 + (int)(0.5 * iAux1)][2] = iK;
                    iMat[iCont3 * 2 + iCont2 + (int)(0.5 * iAux1)][3] = iL;
                    iCont2++;
                };
                iCont4++;
                if(iCont4 == 4) {
                    iCont1 = 0;
                    iCont2 = 0;
                    iCont3++;
                    iCont4 = 0;
                };
                if(iTemp1 < 4) iTemp1++;
                else
                    iTemp1 = 1;
            };
};
iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.5 * iAux1); il++) {

```

```

        vGrava_Out_1(usOpcao, iMat[iI][0], iMat[iI][1], iMat[iI][2], iMat[iI][3], 21);
        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.5 * iAux1), 20);
    };
    for(il = 0; il < (int)(0.5 * iAux1); il++) {
        vGrava_Out_2(usOpcao, iMat[iI + (int)(0.5 * iAux1)][0], iMat[iI + (int)(0.5 * iAux1)][1],
iMat[iI + (int)(0.5 * iAux1)][2], iMat[iI + (int)(0.5 * iAux1)][3], 23);
        iCont2++;
        vMostrar_Barra_3(cNome3, iCont2, (int)(0.5 * iAux1), 22);
    };
}

```

```

void vTipo_3(const int iAux1, const int iAux2, const int iAux3, const int iAux4, const unsigned short
usOpcao, const char *cNome2, const char *cNome3) {

```

```

    int    iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iI = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0;

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.25 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.75 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[iI][0] = -1;
        iMat[iI][1] = -1;
        iMat[iI][2] = -1;
        iMat[iI][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iTemp1 = 1;
    for(il = 0; il < 10; il++)
        for(iJ = 0; iJ < 2; iJ++)
            for(iK = 0; iK < 10; iK++)
                for(iL = 0; iL < 10; iL++) {
                    if((iTemp1 != iAux2) && (iTemp1 != iAux3) && (iTemp1 != iAux4))
{
                        iMat[iCont2][0] = iI;
                        iMat[iCont2][1] = iJ;
                        iMat[iCont2][2] = iK;
                        iMat[iCont2][3] = iL;
                    } else {
                        iMat[iCont2 * 3 + iCont1 + (int)(0.25 * iAux1)][0] = iI;
                        iMat[iCont2 * 3 + iCont1 + (int)(0.25 * iAux1)][1] = iJ;
                        iMat[iCont2 * 3 + iCont1 + (int)(0.25 * iAux1)][2] = iK;
                        iMat[iCont2 * 3 + iCont1 + (int)(0.25 * iAux1)][3] = iL;
                    }
                }
            }
}

```

```

        iCont1++;
    };
    iCont3++;
    if(iCont3 == 4) {
        iCont1 = 0;
        iCont2++;
        iCont3 = 0;
    };
    if(iTemp1 < 4) iTemp1++;
    else
        iTemp1 = 1;
};

iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.25 * iAux1); il++) {
    vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
    iCont1++;
    vMostrar_Barra_2(cNome2, iCont1, (int)(0.25 * iAux1), 20);
};
for(il = 0; il < (int)(0.75 * iAux1); il++) {
    vGrava_Out_2(usOpcao, iMat[il + (int)(0.25 * iAux1)][0], iMat[il + (int)(0.25 * iAux1)][1],
iMat[il + (int)(0.25 * iAux1)][2], iMat[il + (int)(0.25 * iAux1)][3], 23);
    iCont2++;
    vMostrar_Barra_3(cNome3, iCont2, (int)(0.75 * iAux1), 22);
};
}

```

## A.2 Plutão

```

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

```

```

FILE    *fiArqIn,
        *fiArqOut1,
        *fiArqOut2;

```

```

void vApresentacao(void);

```

```

void vObtem_Nome_Arquivo(char *cNome1);

```

```

void vObtem_Tipo_Arquivo(unsigned short *usOpcao);

```

```

void vAbrir_Arquivo_In(const char *cNome1);

```

```

void vAbrir_Arquivo_Out_1(const char *cNome2);

```

```

void vAbrir_Arquivo_Out_2(const char *cNome3);

```

```

void vLimpa_Linha(unsigned short int usLinha);

```

```

void vGrava_Out_1(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha);

void vGrava_Out_2(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha);

void vMostrar_Barra_1(const char *cNome1, const int iCont1, const int iAux1, const unsigned short
usLinha);

void vMostrar_Barra_2(const char *cNome2, const int iCont2, const int iAux1, const unsigned short
usLinha);

void vMostrar_Barra_3(const char *cNome3, const int iCont3, const int iAux1, const unsigned short
usLinha);

void vTipo_0(const int iAux1, const unsigned short usOpcao);

void vTipo_0_Caso_1(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);

void vTipo_0_Caso_2(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);

void vTipo_0_Caso_3(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);

void vTipo_1(const int iAux1, const int iAux2, const unsigned short usOpcao, const char *cNome2, const
char *cNome3);

void vTipo_2(const int iAux1, const int iAux2, const int iAux3, const unsigned short usOpcao, const char
*cNome2, const char *cNome3);

void vTipo_3(const int iAux1, const int iAux2, const int iAux3, const int iAux4, const unsigned short
usOpcao, const char *cNome2, const char *cNome3);

void main(void) {
    char    cNome1[81],
            cNome2[81],
            cNome3[81];
    int     iAux1 = 0,
            iAux2 = 0,
            iAux3 = 0,
            iAux4 = 0,
            iCont1 = 0,
            il = 0,
            iTipo1 = 0,
            iTipo2 = 0;
    unsigned short usOpcao;

    randomize();
    vApresentacao();
    vObtem_Nome_Arquivo(cNome1);

```



```

vObtem_Tipo_Arquivo(&usOpcao);
vAbrir_Arquivo_In(cNome1);
fscanf(fiArqIn, "%d%d", &iCont1, &iAux1);
gotoxy(1, 18);
printf("%s 0.0%%", cNome1);
vLimpa_Linha(20);
vLimpa_Linha(21);
vLimpa_Linha(22);
vLimpa_Linha(23);
vTipo_0(iAux1, usOpcao);
fclose(fiArqOut1);
fclose(fiArqOut2);
for(il = 0; il < iCont1; il++) {
    fscanf(fiArqIn, "%d", &iTipo1);
    vLimpa_Linha(20);
    vLimpa_Linha(21);
    vLimpa_Linha(22);
    vLimpa_Linha(23);
    switch(iTipo1) {
        case 0 :
            fscanf(fiArqIn, "%d%s%s", &iTipo2, cNome2, cNome3);
            switch(iTipo2) {
                case 1 :
                    vTipo_0_Caso_1(iAux1, usOpcao, cNome2, cNome3);
                    break;
                case 2 :
                    vTipo_0_Caso_2(iAux1, usOpcao, cNome2, cNome3);
                    break;
                case 3 :
                    vTipo_0_Caso_3(iAux1, usOpcao, cNome2, cNome3);
                    break;
            };
            break;
        case 1 :
            fscanf(fiArqIn, "%d%s%s", &iAux2, cNome2, cNome3);
            vTipo_1(iAux1, iAux2, usOpcao, cNome2, cNome3);
            break;
        case 2 :
            fscanf(fiArqIn, "%d%d%s%s", &iAux2, &iAux3, cNome2, cNome3);
            vTipo_2(iAux1, iAux2, iAux3, usOpcao, cNome2, cNome3);
            break;
        case 3 :
            fscanf(fiArqIn, "%d%d%d%s%s", &iAux2, &iAux3, &iAux4, cNome2, cNome3);
            vTipo_3(iAux1, iAux2, iAux3, iAux4, usOpcao, cNome2, cNome3);
            break;
    };
    fclose(fiArqOut1);
    fclose(fiArqOut2);
    vMostrar_Barra_1(cNome1, il + 1, iCont1, 18);
};
fclose(fiArqIn);
}

```

```

void vApresentacao(void) {

    clrscr();
    printf("Centro Universitario de Brasilia - UniCEUB\n");
    printf("Faculdade de Ciencias Exatas e Tecnologia - FAET\n");
}

void vObtem_Nome_Arquivo(char *cNome1) {

    printf("Entre com o nome do arquivo gerador de arquivos de treinamento e reconhecimento pela
RNA:\nNome = ");
    scanf("%s", cNome1);
}

void vObtem_Tipo_Arquivo(unsigned short *usOpcao) {

    printf("\nEscolha qual o tipo de figuras sera manipulado:\n1 - Digitos\n2 - Vogais\nOpcao = ");
    scanf("%u", usOpcao);
    if((*usOpcao != 1) && (*usOpcao != 2))
        *usOpcao = 1;
}

void vAbrir_Arquivo_In(const char *cNome1) {

    fiArqIn = fopen(cNome1, "rt");
    if(fiArqIn == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome1);
        exit(1);
    };
}

void vAbrir_Arquivo_Out_1(const char *cNome2) {

    fiArqOut1 = fopen(cNome2, "wt");
    if(fiArqOut1 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome2);
        exit(1);
    };
}

void vAbrir_Arquivo_Out_2(const char *cNome3) {

    fiArqOut2 = fopen(cNome3, "wt");
    if(fiArqOut2 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome3);
        exit(1);
    };
}

void vLimpa_Linha(unsigned short int usLinha) {

    unsigned short usl;

    for(ysl = 0; ysl < 80; ysl++) {

```

```

        gotoxy(usl, usLinha);
        printf(" ");
    };
}

```

```

void vGrava_Out_1(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const unsigned short usLinha) {

```

```

    if(usOpcao == 1) {
        fprintf(fiArqOut1, "%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    } else {
        fprintf(fiArqOut1, "%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    };
}

```

```

void vGrava_Out_2(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const unsigned short usLinha) {

```

```

    if(usOpcao == 1) {
        fprintf(fiArqOut2, "%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    } else {
        fprintf(fiArqOut2, "%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    };
}

```

```

void vMostrar_Barra_1(const char *cNome1, const int iCont1, const int iAux1, const unsigned short usLinha) {

```

```

    int iM;
    float fPorc;

    gotoxy(1, usLinha);
    printf("%s ", cNome1);
    fPorc = (1.0 * iCont1) / iAux1;
    for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
        gotoxy(strlen(cNome1) + 2 + iM, usLinha);
        printf("*");
    };
    gotoxy(strlen(cNome1) + 2 + iM, usLinha);
    printf(" %.1f %%", 100.0*fPorc);
}

```

```

void vMostrar_Barra_2(const char *cNome2, const int iCont2, const int iAux1, const unsigned short usLinha) {

```

```

    int iM;

```

```

float fPorc;

gotoxy(1, usLinha);
printf("%s ", cNome2);
fPorc = (1.0 * iCont2) / iAux1;
for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
    gotoxy(strlen(cNome2) + 2 + iM, usLinha);
    printf("*");
};
gotoxy(strlen(cNome2) + 2 + iM, usLinha);
printf(" %.1f %%", 100.0*fPorc);
}

void vMostrar_Barra_3(const char *cNome3, const int iCont3, const int iAux1, const unsigned short
usLinha) {

    int iM;
    float fPorc;

    gotoxy(1, usLinha);
    printf("%s ", cNome3);
    fPorc = (1.0 * iCont3) / iAux1;
    for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
        gotoxy(strlen(cNome3) + 2 + iM, usLinha);
        printf("*");
    };
    gotoxy(strlen(cNome3) + 2 + iM, usLinha);
    printf(" %.1f %%", 100.0*fPorc);
}

void vTipo_0(const int iAux1, const unsigned short usOpcao) {

    int      iAux2 = 4,
             iCont1 = 0,
             iCont2 = 0,
             iI = 0,
             iJ = 0,
             iK = 0,
             iL = 0,
             iMat[2000][4],
             iTemp1 = 0,
             iTemp2 = 0;

    if(usOpcao == 1) {
        vAbrir_Arquivo_Out_1("DPlut00.txt");
        vAbrir_Arquivo_Out_2("DPlur00.txt");
    } else {
        vAbrir_Arquivo_Out_1("VPlut00.txt");
        vAbrir_Arquivo_Out_2("VPlur00.txt");
    };
    fprintf(fiArqOut1, "%d\n", iAux1);
    fprintf(fiArqOut2, "%d\n", iAux1);
    for(iI = 0; iI < iAux1; iI++) {
        iMat[iI][0] = -1;

```

```

        iMat[iI][1] = -1;
        iMat[iI][2] = -1;
        iMat[iI][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iTemp1 = random((int)(iAux1 / iAux2));
    iTemp2 = random(4);
    for(il = 0; il < 10; il++)
        for(iJ = 0; iJ < 2; iJ++)
            for(iK = 0; iK < 10; iK++)
                for(iL = 0; iL < 10; iL++) {
                    iMat[iTemp1 * 4 + iTemp2][0] = iI;
                    iMat[iTemp1 * 4 + iTemp2][1] = iJ;
                    iMat[iTemp1 * 4 + iTemp2][2] = iK;
                    iMat[iTemp1 * 4 + iTemp2][3] = iL;
                    iCont1++;
                    iCont2++;
                    if((iCont1 == 4) && (iCont2 < iAux1)) {
                        do
                            iTemp1 = random((int)(iAux1 / iAux2));
                            while(iMat[iTemp1 * 4][0] != -1);
                            iCont1 = 0;
                        };
                        if(iCont1 < 4)
                            do
                                iTemp2 = random(4);
                                while(iMat[iTemp1 * 4 + iTemp2][0] != -1);
                            };
                    };
    iCont1 = 0;
    iCont2 = 0;
    for(il = 0; il < iAux1; il++) {
        vGrava_Out_1(usOpcao, iMat[iI][0], iMat[iI][1], iMat[iI][2], iMat[iI][3], 21);
        iCont1++;
        if(usOpcao == 1)
            vMostrar_Barra_2("DPlut00.txt", iCont1, iAux1, 20);
        else
            vMostrar_Barra_2("VPlut00.txt", iCont1, iAux1, 20);
    };
    for(il = 0; il < iAux1; il++) {
        vGrava_Out_2(usOpcao, iMat[iI][0], iMat[iI][1], iMat[iI][2], iMat[iI][3], 23);
        iCont2++;
        if(usOpcao == 1)
            vMostrar_Barra_3("DPlur00.txt", iCont2, iAux1, 22);
        else
            vMostrar_Barra_3("VPlur00.txt", iCont2, iAux1, 22);
    };
}

void vTipo_0_Caso_1(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {
    int    iAux2 = 3,
           iAux3 = 0,

```

```

        iCont1 = 0,
        iCont2 = 0,
        iCont3 = 0,
        il = 0,
        iJ = 0,
        iK = 0,
        iL = 0,
        iMat[2000][4],
        iTemp1 = 0,
        iTemp2 = 0,
        iTemp3 = 0;

vAbrir_Arquivo_Out_1(cNome2);
vAbrir_Arquivo_Out_2(cNome3);
fprintf(fiArqOut1, "%d\n", (int)(0.75 * iAux1));
fprintf(fiArqOut2, "%d\n", (int)(0.25 * iAux1));
for(il = 0; il < iAux1; il++) {
    iMat[il][0] = -1;
    iMat[il][1] = -1;
    iMat[il][2] = -1;
    iMat[il][3] = -1;
};
iCont1 = 0;
iCont2 = 0;
iCont3 = 0;
iTemp1 = 1;
iTemp2 = random((int)((int)(0.75 * iAux1) / iAux2));
iTemp3 = random(3);
iAux3 = random(4)+1;
for(il = 0; il < 10; il++)
    for(iJ = 0; iJ < 2; iJ++)
        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                if(iTemp1 != iAux3) {
                    iMat[iTemp2 * 3 + iTemp3][0] = il;
                    iMat[iTemp2 * 3 + iTemp3][1] = iJ;
                    iMat[iTemp2 * 3 + iTemp3][2] = iK;
                    iMat[iTemp2 * 3 + iTemp3][3] = iL;
                    iCont1++;
                } else {
                    iMat[iTemp2 + (int)(0.75 * iAux1)][0] = il;
                    iMat[iTemp2 + (int)(0.75 * iAux1)][1] = iJ;
                    iMat[iTemp2 + (int)(0.75 * iAux1)][2] = iK;
                    iMat[iTemp2 + (int)(0.75 * iAux1)][3] = iL;
                };
                iCont2++;
                iCont3++;
                if((iCont2 == 4) && (iCont3 < iAux1)) {
                    do
                        iTemp2 = random((int)((int)(0.75 * iAux1) /
iAux2));

                    while(iMat[iTemp2 * 3][0] != -1);
                    iCont1 = 0;
                    iCont2 = 0;

```

```

        };
        if(iCont1 < 3)
            do
                iTemp3 = random(3);
                while(iMat[iTemp2 * 3 + iTemp3][0] != -1);
            if(iTemp1 < 4) iTemp1++;
            else {
                iTemp1 = 1;
                iAux3 = random(4)+1;
            };
    };

    iCont1 = 0;
    iCont2 = 0;
    for(il = 0; il < (int)(0.75 * iAux1); il++) {
        vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.75 * iAux1), 20);
    };
    for(il = 0; il < (int)(0.25 * iAux1); il++) {
        vGrava_Out_2(usOpcao, iMat[il + (int)(0.75 * iAux1)][0], iMat[il + (int)(0.75 * iAux1)][1],
iMat[il + (int)(0.75 * iAux1)][2], iMat[il + (int)(0.75 * iAux1)][3], 23);
        iCont2++;
        vMostrar_Barra_3(cNome3, iCont2, (int)(0.25 * iAux1), 22);
    };
}

void vTipo_0_Caso_2(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {

    int    iAux2 = 2,
           iAux3 = 0,
           iAux4 = 0,
           iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iCont4 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0,
           iTemp2 = 0,
           iTemp3 = 0,
           iTemp4 = 0;

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.5 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.5 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;

```

```

        iMat[iI][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iCont4 = 0;
    iTemp1 = 1;
    iTemp2 = random((int)((int)(0.5 * iAux1) / iAux2));
    iTemp3 = random(2);
    iTemp4 = random(2);
    iAux3 = random(4)+1;
    do
        iAux4 = random(4) + 1;
    while(iAux4 == iAux3);
    for(il = 0; il < 10; il++)
        for(iJ = 0; iJ < 2; iJ++)
            for(iK = 0; iK < 10; iK++)
                for(iL = 0; iL < 10; iL++) {
                    if((iTemp1 != iAux3) && (iTemp1 != iAux4)) {
                        iMat[iTemp2 * 2 + iTemp3][0] = iI;
                        iMat[iTemp2 * 2 + iTemp3][1] = iJ;
                        iMat[iTemp2 * 2 + iTemp3][2] = iK;
                        iMat[iTemp2 * 2 + iTemp3][3] = iL;
                        iCont1++;
                    } else {
                        iMat[iTemp2 * 2 + iTemp4 + (int)(0.5 * iAux1)][0] = iI;
                        iMat[iTemp2 * 2 + iTemp4 + (int)(0.5 * iAux1)][1] = iJ;
                        iMat[iTemp2 * 2 + iTemp4 + (int)(0.5 * iAux1)][2] = iK;
                        iMat[iTemp2 * 2 + iTemp4 + (int)(0.5 * iAux1)][3] = iL;
                        iCont2++;
                    }
                };
    iCont3++;
    iCont4++;
    if((iCont3 == 4) && (iCont4 < iAux1)) {
        do
            iTemp2 = random((int)((int)(0.5 * iAux1) / iAux2));
            while(iMat[iTemp2 * 2][0] != -1);
            iCont1 = 0;
            iCont2 = 0;
            iCont3 = 0;
        };
        if(iCont1 < 2)
            do
                iTemp3 = random(2);
                while(iMat[iTemp2 * 2 + iTemp3][0] != -1);
            if(iCont2 < 2)
                do
                    iTemp4 = random(2);
                    while(iMat[iTemp2 * 2 + iTemp4 + (int)(0.5 * iAux1)][0] != -
1);
        if(iTemp1 < 4) iTemp1++;
        else {
            iTemp1 = 1;
            iAux3 = random(4) + 1;

```



```

do
    iAux4 = random(4) + 1;
while(iAux4 == iAux3);
};
};

iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
    iCont1++;
    vMostrar_Barra_2(cNome2, iCont1, (int)(0.5 * iAux1), 20);
};
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_2(usOpcao, iMat[il + (int)(0.5 * iAux1)][0], iMat[il + (int)(0.5 * iAux1)][1],
iMat[il + (int)(0.5 * iAux1)][2], iMat[il + (int)(0.5 * iAux1)][3], 23);
    iCont2++;
    vMostrar_Barra_3(cNome3, iCont2, (int)(0.5 * iAux1), 22);
};
}

void vTipo_0_Caso_3(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {
    int    iAux2 = 1,
           iAux3 = 0,
           iAux4 = 0,
           iAux5 = 0,
           iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0,
           iTemp2 = 0,
           iTemp3 = 0;

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.25 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.75 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
        iMat[il][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iTemp1 = 1;
    iTemp2 = random(((int)((int)(0.25 * iAux1) / iAux2)));

```

```

iTemp3 = random(3);
iAux3 = random(4)+1;
do
    iAux4 = random(4) + 1;
while(iAux4 == iAux3);
do
    iAux5 = random(4) + 1;
while((iAux5 == iAux3) || (iAux5 == iAux4));
for(il = 0; il < 10; il++)
    for(iJ = 0; iJ < 2; iJ++)
        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                if((iTemp1 != iAux3) && (iTemp1 != iAux4) && (iTemp1 != iAux5))
                    iMat[iTemp2][0] = il;
                    iMat[iTemp2][1] = iJ;
                    iMat[iTemp2][2] = iK;
                    iMat[iTemp2][3] = iL;
                } else {
                    iMat[iTemp2 * 3 + iTemp3 + (int)(0.25 * iAux1)][0] = il;
                    iMat[iTemp2 * 3 + iTemp3 + (int)(0.25 * iAux1)][1] = iJ;
                    iMat[iTemp2 * 3 + iTemp3 + (int)(0.25 * iAux1)][2] = iK;
                    iMat[iTemp2 * 3 + iTemp3 + (int)(0.25 * iAux1)][3] = iL;
                    iCont1++;
                };
                iCont2++;
                iCont3++;
                if((iCont2 == 4) && (iCont3 < iAux1)) {
                    do
                        iTemp2 = random((int)((int)(0.25 * iAux1) /
iAux2));

                        while(iMat[iTemp2][0] != -1);
                        iCont1 = 0;
                        iCont2 = 0;
                    };
                    if(iCont1 < 3)
                        do
                            iTemp3 = random(3);
                            while(iMat[iTemp2 * 3 + iTemp3 + (int)(0.25 * iAux1)][0] !=
-1);

                            if(iTemp1 < 4) iTemp1++;
                            else {
                                iTemp1 = 1;
                                iAux3 = random(4) + 1;
                                do
                                    iAux4 = random(4) + 1;
                                while(iAux4 == iAux3);
                                do
                                    iAux5 = random(4) + 1;
                                while((iAux5 == iAux3) || (iAux5 == iAux4));
                            };
                        };
                iCont1 = 0;
                iCont2 = 0;

```

```

        for(il = 0; il < (int)(0.25 * iAux1); il++) {
            vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
            iCont1++;
            vMostrar_Barra_2(cNome2, iCont1, (int)(0.25 * iAux1), 20);
        };
        for(il = 0; il < (int)(0.75 * iAux1); il++) {
            vGrava_Out_2(usOpcao, iMat[il + (int)(0.25 * iAux1)][0], iMat[il + (int)(0.25 * iAux1)][1],
iMat[il + (int)(0.25 * iAux1)][2], iMat[il + (int)(0.25 * iAux1)][3], 23);
            iCont2++;
            vMostrar_Barra_3(cNome3, iCont2, (int)(0.75 * iAux1), 22);
        };
    }

```

```

void vTipo_1(const int iAux1, const int iAux2, const unsigned short usOpcao, const char *cNome2, const
char *cNome3) {

```

```

    int        iAux3 = 3,
                iCont1 = 0,
                iCont2 = 0,
                iCont3 = 0,
                il = 0,
                iJ = 0,
                iK = 0,
                iL = 0,
                iMat[2000][4],
                iTemp1 = 0,
                iTemp2 = 0,
                iTemp3 = 0;

```

```

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.75 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.25 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
        iMat[il][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iTemp1 = 1;
    iTemp2 = random((int)((int)(0.75 * iAux1) / iAux3));
    iTemp3 = random(3);
    for(il = 0; il < 10; il++)
        for(iJ = 0; iJ < 2; iJ++)
            for(iK = 0; iK < 10; iK++)
                for(iL = 0; iL < 10; iL++) {
                    if(iTemp1 != iAux2) {
                        iMat[iTemp2 * 3 + iTemp3][0] = il;
                        iMat[iTemp2 * 3 + iTemp3][1] = iJ;
                        iMat[iTemp2 * 3 + iTemp3][2] = iK;
                        iMat[iTemp2 * 3 + iTemp3][3] = iL;
                    }
                }
    }

```

```

        iCont1++;
    } else {
        iMat[iTemp2 + (int)(0.75 * iAux1)][0] = iI;
        iMat[iTemp2 + (int)(0.75 * iAux1)][1] = iJ;
        iMat[iTemp2 + (int)(0.75 * iAux1)][2] = iK;
        iMat[iTemp2 + (int)(0.75 * iAux1)][3] = iL;
    };
    iCont2++;
    iCont3++;
    if((iCont2 == 4) && (iCont3 < iAux1)) {
        do
            iTemp2 = random((int)((int)(0.75 * iAux1) /
iAux3));

            while(iMat[iTemp2 * 3][0] != -1);
            iCont1 = 0;
            iCont2 = 0;
        };
        if(iCont1 < 3)
            do
                iTemp3 = random(3);
                while(iMat[iTemp2 * 3 + iTemp3][0] != -1);
            if(iTemp1 < 4) iTemp1++;
            else iTemp1 = 1;
        };
    iCont1 = 0;
    iCont2 = 0;
    for(il = 0; il < (int)(0.75 * iAux1); il++) {
        vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.75 * iAux1), 20);
    };
    for(il = 0; il < (int)(0.25 * iAux1); il++) {
        vGrava_Out_2(usOpcao, iMat[il + (int)(0.75 * iAux1)][0], iMat[il + (int)(0.75 * iAux1)][1],
iMat[il + (int)(0.75 * iAux1)][2], iMat[il + (int)(0.75 * iAux1)][3], 23);
        iCont2++;
        vMostrar_Barra_3(cNome3, iCont2, (int)(0.25 * iAux1), 22);
    };
}

void vTipo_2(const int iAux1, const int iAux2, const int iAux3, const unsigned short usOpcao, const char
*cNome2, const char *cNome3) {

    int    iAux4 = 2,
           iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iCont4 = 0,
           iI = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0,
           iTemp2 = 0,

```

```

        iTemp3 = 0;
        iTemp4 = 0;

vAbrir_Arquivo_Out_1(cNome2);
vAbrir_Arquivo_Out_2(cNome3);
fprintf(fiArqOut1, "%d\n", (int)(0.5 * iAux1));
fprintf(fiArqOut2, "%d\n", (int)(0.5 * iAux1));
for(il = 0; il < iAux1; il++) {
    iMat[ii][0] = -1;
    iMat[ii][1] = -1;
    iMat[ii][2] = -1;
    iMat[ii][3] = -1;
};
iCont1 = 0;
iCont2 = 0;
iCont3 = 0;
iCont4 = 0;
iTemp1 = 1;
iTemp2 = random((int)((int)(0.5 * iAux1) / iAux4));
iTemp3 = random(2);
iTemp4 = random(2);
for(il = 0; il < 10; il++)
    for(iJ = 0; iJ < 2; iJ++)
        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                if((iTemp1 != iAux2) && (iTemp1 != iAux3)) {
                    iMat[iTemp2 * 2 + iTemp3][0] = il;
                    iMat[iTemp2 * 2 + iTemp3][1] = iJ;
                    iMat[iTemp2 * 2 + iTemp3][2] = iK;
                    iMat[iTemp2 * 2 + iTemp3][3] = iL;
                    iCont1++;
                } else {
                    iMat[iTemp2 * 2 + iTemp4 + (int)(0.5 * iAux1)][0] = il;
                    iMat[iTemp2 * 2 + iTemp4 + (int)(0.5 * iAux1)][1] = iJ;
                    iMat[iTemp2 * 2 + iTemp4 + (int)(0.5 * iAux1)][2] = iK;
                    iMat[iTemp2 * 2 + iTemp4 + (int)(0.5 * iAux1)][3] = iL;
                    iCont2++;
                }
            };
iCont3++;
iCont4++;
if((iCont3 == 4) && (iCont4 < iAux1)) {
    do
        iTemp2 = random((int)((int)(0.5 * iAux1) / iAux4));
        while(iMat[iTemp2 * 2][0] != -1);
        iCont1 = 0;
        iCont2 = 0;
        iCont3 = 0;
    };
if(iCont1 < 2)
    do
        iTemp3 = random(2);
        while(iMat[iTemp2 * 2 + iTemp3][0] != -1);
if(iCont2 < 2)
    do

```

```

        iTemp4 = random(2);
        while(iMat[iTemp2 * 2 + iTemp4 + (int)(0.5 * iAux1)][0] != -
1);

        if(iTemp1 < 4) iTemp1++;
        else iTemp1 = 1;

    };

    iCont1 = 0;
    iCont2 = 0;
    for(il = 0; il < (int)(0.5 * iAux1); il++) {
        vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.5 * iAux1), 20);
    };
    for(il = 0; il < (int)(0.5 * iAux1); il++) {
        vGrava_Out_2(usOpcao, iMat[il + (int)(0.5 * iAux1)][0], iMat[il + (int)(0.5 * iAux1)][1],
iMat[il + (int)(0.5 * iAux1)][2], iMat[il + (int)(0.5 * iAux1)][3], 23);
        iCont2++;
        vMostrar_Barra_3(cNome3, iCont2, (int)(0.5 * iAux1), 22);
    };
}

```

```

void vTipo_3(const int iAux1, const int iAux2, const int iAux3, const int iAux4, const unsigned short
usOpcao, const char *cNome2, const char *cNome3) {

```

```

    int    iAux5 = 1,
           iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0,
           iTemp2 = 0,
           iTemp3 = 0;

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.25 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.75 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
        iMat[il][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iTemp1 = 1;
    iTemp2 = random((int)((int)(0.25 * iAux1) / iAux5));
    iTemp3 = random(3);
    for(il = 0; il < 10; il++)

```

```

for(iJ = 0; iJ < 2; iJ++)
    for(iK = 0; iK < 10; iK++)
        for(iL = 0; iL < 10; iL++) {
            if((iTemp1 != iAux2) && (iTemp1 != iAux3) && (iTemp1 != iAux4))
            {
                iMat[iTemp2][0] = iL;
                iMat[iTemp2][1] = iJ;
                iMat[iTemp2][2] = iK;
                iMat[iTemp2][3] = iL;
            } else {
                iMat[iTemp2 * 3 + iTemp3 + (int)(0.25 * iAux1)][0] = iL;
                iMat[iTemp2 * 3 + iTemp3 + (int)(0.25 * iAux1)][1] = iJ;
                iMat[iTemp2 * 3 + iTemp3 + (int)(0.25 * iAux1)][2] = iK;
                iMat[iTemp2 * 3 + iTemp3 + (int)(0.25 * iAux1)][3] = iL;
                iCont1++;
            };
            iCont2++;
            iCont3++;
            if((iCont2 == 4) && (iCont3 < iAux1)) {
                do
                    iTemp2 = random((int)((int)(0.25 * iAux1) /
iAux5));

                    while(iMat[iTemp2][0] != -1);
                    iCont1 = 0;
                    iCont2 = 0;
                };
                if(iCont1 < 3)
                do
                    iTemp3 = random(3);
                    while(iMat[iTemp2 * 3 + iTemp3 + (int)(0.25 * iAux1)][0] !=
-1);

                    if(iTemp1 < 4) iTemp1++;
                    else iTemp1 = 1;
                };
            }
            iCont1 = 0;
            iCont2 = 0;
            for(il = 0; il < (int)(0.25 * iAux1); il++) {
                vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
                iCont1++;
                vMostrar_Barra_2(cNome2, iCont1, (int)(0.25 * iAux1), 20);
            };
            for(il = 0; il < (int)(0.75 * iAux1); il++) {
                vGrava_Out_2(usOpcao, iMat[il + (int)(0.25 * iAux1)][0], iMat[il + (int)(0.25 * iAux1)][1],
iMat[il + (int)(0.25 * iAux1)][2], iMat[il + (int)(0.25 * iAux1)][3], 23);
                iCont2++;
                vMostrar_Barra_3(cNome3, iCont2, (int)(0.75 * iAux1), 22);
            };
        }
}

```

## A.3 Netuno

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

FILE    *fiArqIn,
        *fiArqOut1,
        *fiArqOut2;

void vApresentacao(void);

void vObtem_Nome_Arquivo(char *cNome1);

void vObtem_Tipo_Arquivo(unsigned short *usOpcao);

void vAbrir_Arquivo_In(const char *cNome1);

void vAbrir_Arquivo_Out_1(const char *cNome2);

void vAbrir_Arquivo_Out_2(const char *cNome3);

void vLimpa_Linha(unsigned short int usLinha);

void vGrava_Out_1(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha);

void vGrava_Out_2(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha);

void vMostrar_Barra_1(const char *cNome1, const int iCont1, const int iAux1, const unsigned short
usLinha);

void vMostrar_Barra_2(const char *cNome2, const int iCont2, const int iAux1, const unsigned short
usLinha);

void vMostrar_Barra_3(const char *cNome3, const int iCont3, const int iAux1, const unsigned short
usLinha);

void vTipo_0(const int iAux1, const unsigned short usOpcao);

void vTipo_0_Caso_1(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);

void vTipo_0_Caso_2(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);

void vTipo_0_Caso_3(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);
```



```
void vTipo_1(const int iAux1, const int iAux2, const unsigned short usOpcao, const char *cNome2, const char *cNome3);
```

```
void vTipo_2(const int iAux1, const int iAux2, const int iAux3, const unsigned short usOpcao, const char *cNome2, const char *cNome3);
```

```
void vTipo_3(const int iAux1, const int iAux2, const int iAux3, const int iAux4, const unsigned short usOpcao, const char *cNome2, const char *cNome3);
```

```
void main(void) {
```

```
    char    cNome1[81],
            cNome2[81],
            cNome3[81];
```

```
    int      iAux1 = 0,
            iAux2 = 0,
            iAux3 = 0,
            iAux4 = 0,
            iCont1 = 0,
            il = 0,
            iTipo1 = 0,
            iTipo2 = 0;
```

```
    unsigned short usOpcao;
```

```
    randomize();
```

```
    vApresentacao();
```

```
    vObtem_Nome_Arquivo(cNome1);
```

```
    vObtem_Tipo_Arquivo(&usOpcao);
```

```
    vAbrir_Arquivo_In(cNome1);
```

```
    fscanf(fiArqIn, "%d%d", &iCont1, &iAux1);
```

```
    gotoxy(1, 18);
```

```
    printf("%s 0.0%%", cNome1);
```

```
    vLimpa_Linha(20);
```

```
    vLimpa_Linha(21);
```

```
    vLimpa_Linha(22);
```

```
    vLimpa_Linha(23);
```

```
    vTipo_0(iAux1, usOpcao);
```

```
    fclose(fiArqOut1);
```

```
    fclose(fiArqOut2);
```

```
    for(il = 0; il < iCont1; il++) {
        fscanf(fiArqIn, "%d", &iTipo1);
```

```
        vLimpa_Linha(20);
```

```
        vLimpa_Linha(21);
```

```
        vLimpa_Linha(22);
```

```
        vLimpa_Linha(23);
```

```
        switch(iTipo1) {
```

```
            case 0 :
```

```
                fscanf(fiArqIn, "%d%s%s", &iTipo2, cNome2, cNome3);
```

```
                switch(iTipo2) {
```

```
                    case 1 :
```

```
                        vTipo_0_Caso_1(iAux1, usOpcao, cNome2, cNome3);
```

```
                    break;
```

```
                    case 2 :
```

```
                        vTipo_0_Caso_2(iAux1, usOpcao, cNome2, cNome3);
```

```

                break;
                case 3 :
                    vTipo_0_Caso_3(iAux1, usOpcao, cNome2, cNome3);
                break;
            };
        break;
        case 1 :
            fscanf(fiArqIn,"%d%s%s",&iAux2,cNome2,cNome3);
            vTipo_1(iAux1, iAux2, usOpcao, cNome2, cNome3);
        break;
        case 2 :
            fscanf(fiArqIn,"%d%d%s%s",&iAux2,&iAux3,cNome2,cNome3);
            vTipo_2(iAux1, iAux2, iAux3, usOpcao, cNome2, cNome3);
        break;
        case 3 :

            fscanf(fiArqIn,"%d%d%d%s%s",&iAux2,&iAux3,&iAux4,cNome2,cNome3);
            vTipo_3(iAux1, iAux2, iAux3, iAux4, usOpcao, cNome2, cNome3);
        break;
    };
    fclose(fiArqOut1);
    fclose(fiArqOut2);
    vMostrar_Barra_1(cNome1, il + 1, iCont1, 18);
};
fclose(fiArqIn);
}

void vApresentacao(void) {

    clrscr();
    printf("Centro Universitario de Brasilia - UniCEUB\n");
    printf("Faculdade de Ciencias Exatas e Tecnologia - FAET\n");
}

void vObtem_Nome_Arquivo(char *cNome1) {

    printf("Entre com o nome do arquivo gerador de arquivos de treinamento e reconhecimento pela
RNA:\nNome = ");
    scanf("%s", cNome1);
}

void vObtem_Tipo_Arquivo(unsigned short *usOpcao) {

    printf("\nEscolha qual o tipo de figuras sera manipulado:\n1 - Digitos\n2 - Vogais\nOpcao = ");
    scanf("%u", usOpcao);
    if((*usOpcao != 1) && (*usOpcao != 2))
        *usOpcao = 1;
}

void vAbrir_Arquivo_In(const char *cNome1) {

    fiArqIn = fopen(cNome1, "rt");
    if(fiArqIn == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome1);
    }
}

```

```

        exit(1);
    };
}

void vAbrir_Arquivo_Out_1(const char *cNome2) {

    fiArqOut1 = fopen(cNome2,"wt");
    if(fiArqOut1 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s",cNome2);
        exit(1);
    };
}

void vAbrir_Arquivo_Out_2(const char *cNome3) {

    fiArqOut2 = fopen(cNome3,"wt");
    if(fiArqOut2 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s",cNome3);
        exit(1);
    };
}

void vLimpa_Linha(unsigned short int usLinha) {

    unsigned short usl;

    for(ysl = 0; ysl < 80; ysl++) {
        gotoxy(ysl, usLinha);
        printf(" ");
    };
}

void vGrava_Out_1(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha) {

    if(usOpcao == 1) {
        fprintf(fiArqOut1, "%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    } else {
        fprintf(fiArqOut1, "%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    };
}

void vGrava_Out_2(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha) {

    if(usOpcao == 1) {
        fprintf(fiArqOut2, "%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    } else {

```

```

        fprintf(fiArqOut2,"%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    };
}

void vMostrar_Barra_1(const char *cNome1, const int iCont1, const int iAux1, const unsigned short
usLinha) {

    int iM;
    float fPorc;

    gotoxy(1, usLinha);
    printf("%s ", cNome1);
    fPorc = (1.0 * iCont1) / iAux1;
    for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
        gotoxy(strlen(cNome1) + 2 + iM, usLinha);
        printf("**");
    };
    gotoxy(strlen(cNome1) + 2 + iM, usLinha);
    printf(" %.1f %%", 100.0*fPorc);
}

void vMostrar_Barra_2(const char *cNome2, const int iCont2, const int iAux1, const unsigned short
usLinha) {

    int iM;
    float fPorc;

    gotoxy(1, usLinha);
    printf("%s ", cNome2);
    fPorc = (1.0 * iCont2) / iAux1;
    for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
        gotoxy(strlen(cNome2) + 2 + iM, usLinha);
        printf("**");
    };
    gotoxy(strlen(cNome2) + 2 + iM, usLinha);
    printf(" %.1f %%", 100.0*fPorc);
}

void vMostrar_Barra_3(const char *cNome3, const int iCont3, const int iAux1, const unsigned short
usLinha) {

    int iM;
    float fPorc;

    gotoxy(1, usLinha);
    printf("%s ", cNome3);
    fPorc = (1.0 * iCont3) / iAux1;
    for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
        gotoxy(strlen(cNome3) + 2 + iM, usLinha);
        printf("**");
    };
    gotoxy(strlen(cNome3) + 2 + iM, usLinha);

```

```

        printf(" %.1f %%",100.0*fPorc);
    }

void vTipo_0(const int iAux1, const unsigned short usOpcao) {

    int      iCont1 = 0,
             iCont2 = 0,
             iCont3 = 0,
             il = 0,
             iJ = 0,
             iK = 0,
             iL = 0,
             iMat[2000][4],
             iTemp1 = 0;

    if(usOpcao == 1) {
        vAbrir_Arquivo_Out_1("DNett00.txt");
        vAbrir_Arquivo_Out_2("DNetr00.txt");
    } else {
        vAbrir_Arquivo_Out_1("VNett00.txt");
        vAbrir_Arquivo_Out_2("VNetr00.txt");
    };
    fprintf(fiArqOut1, "%d\n", iAux1);
    fprintf(fiArqOut2, "%d\n", iAux1);
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
        iMat[il][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iTemp1 = random(4);
    for(iJ = 0; iJ < 2; iJ++)
        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                for(il = 0; il < 10; il++) {
                    iMat[iCont1 * 40 + iTemp1 * 10 + il][0] = il;
                    iMat[iCont1 * 40 + iTemp1 * 10 + il][1] = iJ;
                    iMat[iCont1 * 40 + iTemp1 * 10 + il][2] = iK;
                    iMat[iCont1 * 40 + iTemp1 * 10 + il][3] = iL;
                    iCont2++;
                    iCont3++;
                };
                if(iCont2 == 40) {
                    iCont1++;
                    iCont2 = 0;
                };
                if(iCont3 < iAux1)
                    do
                        iTemp1 = random(4);
                    while(iMat[iCont1 * 40 + iTemp1 * 10][0] != -1);
            };
};

```

```

iCont1 = 0;
iCont2 = 0;
for(il = 0; il < iAux1; il++) {
    vGrava_Out_1(usOpcao, iMat[i][0], iMat[i][1], iMat[i][2], iMat[i][3], 21);
    iCont1++;
    if(usOpcao == 1)
        vMostrar_Barra_2("DNett00.txt", iCont1, iAux1, 20);
    else
        vMostrar_Barra_2("VNett00.txt", iCont1, iAux1, 20);
};
for(il = 0; il < iAux1; il++) {
    vGrava_Out_2(usOpcao, iMat[i][0], iMat[i][1], iMat[i][2], iMat[i][3], 23);
    iCont2++;
    if(usOpcao == 1)
        vMostrar_Barra_3("DNetr00.txt", iCont2, iAux1, 22);
    else
        vMostrar_Barra_3("VNetr00.txt", iCont2, iAux1, 22);
};
}

void vTipo_0_Caso_1(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {
    int    iAux2 = 0,
           iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iCont4 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0,
           iTemp2 = 0;

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.75 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.25 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[i][0] = -1;
        iMat[i][1] = -1;
        iMat[i][2] = -1;
        iMat[i][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iCont4 = 0;
    iTemp1 = 1;
    iTemp2 = random(3);
    iAux2 = random(4)+1;
    for(iJ = 0; iJ < 2; iJ++)

```

```

for(iK = 0; iK < 10; iK++)
    for(iL = 0; iL < 10; iL++) {
        for(il = 0; il < 10; il++) {
            if(iTemp1 != iAux2) {
                iMat[iCont1 * 30 + iTemp2 * 10 + il][0] = il;
                iMat[iCont1 * 30 + iTemp2 * 10 + il][1] = iJ;
                iMat[iCont1 * 30 + iTemp2 * 10 + il][2] = iK;
                iMat[iCont1 * 30 + iTemp2 * 10 + il][3] = iL;
                iCont2++;
            } else {
                iMat[iCont1 * 10 + il + (int)(0.75 * iAux1)][0] = il;
                iMat[iCont1 * 10 + il + (int)(0.75 * iAux1)][1] = iJ;
                iMat[iCont1 * 10 + il + (int)(0.75 * iAux1)][2] = iK;
                iMat[iCont1 * 10 + il + (int)(0.75 * iAux1)][3] = iL;
            };
            iCont3++;
            iCont4++;
        };
        if(iCont3 == 40) {
            iCont1++;
            iCont2 = 0;
            iCont3 = 0;
        };
        if((iCont2 != 30) && (iCont4 < iAux1))
            do
                iTemp2 = random(3);
                while (iMat[iCont1 * 30 + iTemp2 * 10][0] != -1);
            if(iTemp1 < 4) iTemp1++;
            else {
                iTemp1 = 1;
                iAux2 = random(4)+1;
            };
    };
iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.75 * iAux1); il++) {
    vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
    iCont1++;
    vMostrar_Barra_2(cNome2, iCont1, (int)(0.75 * iAux1), 20);
};
for(il = 0; il < (int)(0.25 * iAux1); il++) {
    vGrava_Out_2(usOpcao, iMat[il + (int)(0.75 * iAux1)][0], iMat[il + (int)(0.75 * iAux1)][1],
iMat[il + (int)(0.75 * iAux1)][2], iMat[il + (int)(0.75 * iAux1)][3], 23);
    iCont2++;
    vMostrar_Barra_3(cNome3, iCont2, (int)(0.25 * iAux1), 22);
};
}

void vTipo_0_Caso_2(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {

    int    iAux2 = 0,
           iAux3 = 0,
           iCont1 = 0,

```

```

        iCont2 = 0,
        iCont3 = 0,
        iCont4 = 0,
        iCont5 = 0,
        il = 0,
        iJ = 0,
        iK = 0,
        iL = 0,
        iMat[2000][4],
        iTemp1 = 0,
        iTemp2 = 0,
        iTemp3 = 0;

vAbrir_Arquivo_Out_1(cNome2);
vAbrir_Arquivo_Out_2(cNome3);
fprintf(fiArqOut1, "%d\n", (int)(0.5 * iAux1));
fprintf(fiArqOut2, "%d\n", (int)(0.5 * iAux1));
for(il = 0; il < iAux1; il++) {
    iMat[iJ][0] = -1;
    iMat[iJ][1] = -1;
    iMat[iJ][2] = -1;
    iMat[iJ][3] = -1;
};
iCont1 = 0;
iCont2 = 0;
iCont3 = 0;
iCont4 = 0;
iCont5 = 0;
iTemp1 = 1;
iTemp2 = random(2);
iTemp3 = random(2);
iAux2 = random(4)+1;
do
    iAux3 = random(4) + 1;
while(iAux3 == iAux2);
for(iJ = 0; iJ < 2; iJ++)
    for(iK = 0; iK < 10; iK++)
        for(iL = 0; iL < 10; iL++) {
            for(il = 0; il < 10; il++) {
                if((iTemp1 != iAux2) && (iTemp1 != iAux3)) {
                    iMat[iCont1 * 20 + iTemp2 * 10 + il][0] = il;
                    iMat[iCont1 * 20 + iTemp2 * 10 + il][1] = iJ;
                    iMat[iCont1 * 20 + iTemp2 * 10 + il][2] = iK;
                    iMat[iCont1 * 20 + iTemp2 * 10 + il][3] = iL;
                    iCont2++;
                } else {
                    iMat[iCont1 * 20 + iTemp3 * 10 + il + (int)(0.5 * iAux1)][0]
= il;
                    iMat[iCont1 * 20 + iTemp3 * 10 + il + (int)(0.5 * iAux1)][1]
= iJ;
                    iMat[iCont1 * 20 + iTemp3 * 10 + il + (int)(0.5 * iAux1)][2]
= iK;
                    iMat[iCont1 * 20 + iTemp3 * 10 + il + (int)(0.5 * iAux1)][3]
= iL;

```



```

        iCont3++;
    };
    iCont4++;
    iCont5++;
};
if(iCont4 == 40) {
    iCont1++;
    iCont2 = 0;
    iCont3 = 0;
    iCont4 = 0;
};
if((iTemp1 != iAux2) && (iTemp1 != iAux3)) {
    if((iCont2 != 20) && (iCont5 < iAux1))
        do
            iTemp2 = random(2);
            while (iMat[iCont1 * 20 + iTemp2 * 10][0] != -1);
} else {
    if((iCont3 != 20) && (iCont5 < iAux1))
        do
            iTemp3 = random(2);
            while (iMat[iCont1 * 20 + iTemp3 * 10 + (int)(0.5 *
iAux1)][0] != -1);
};
if(iTemp1 < 4) iTemp1++;
else {
    iTemp1 = 1;
    iAux2 = random(4) + 1;
    do
        iAux3 = random(4) + 1;
        while(iAux3 == iAux2);
};
};

iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
    iCont1++;
    vMostrar_Barra_2(cNome2, iCont1, (int)(0.5 * iAux1), 20);
};
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_2(usOpcao, iMat[il + (int)(0.5 * iAux1)][0], iMat[il + (int)(0.5 * iAux1)][1],
iMat[il + (int)(0.5 * iAux1)][2], iMat[il + (int)(0.5 * iAux1)][3], 23);
    iCont2++;
    vMostrar_Barra_3(cNome3, iCont2, (int)(0.5 * iAux1), 22);
};
}

void vTipo_0_Caso_3(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {
    int    iAux2 = 0,
           iAux3 = 0,
           iAux4 = 0,
           iCont1 = 0,

```

```

        iCont2 = 0,
        iCont3 = 0,
        iCont4 = 0,
        il = 0,
        iJ = 0,
        iK = 0,
        iL = 0,
        iMat[2000][4],
        iTemp1 = 0,
        iTemp2 = 0;

vAbrir_Arquivo_Out_1(cNome2);
vAbrir_Arquivo_Out_2(cNome3);
fprintf(fiArqOut1, "%d\n", (int)(0.25 * iAux1));
fprintf(fiArqOut2, "%d\n", (int)(0.75 * iAux1));
for(il = 0; il < iAux1; il++) {
    iMat[il][0] = -1;
    iMat[il][1] = -1;
    iMat[il][2] = -1;
    iMat[il][3] = -1;
};
iCont1 = 0;
iCont2 = 0;
iCont3 = 0;
iCont4 = 0;
iTemp1 = 1;
iTemp2 = random(3);
iAux2 = random(4)+1;
do
    iAux3 = random(4) + 1;
while(iAux3 == iAux2);
do
    iAux4 = random(4) + 1;
while((iAux4 == iAux2) || (iAux4 == iAux3));
for(iJ = 0; iJ < 2; iJ++)
    for(iK = 0; iK < 10; iK++)
        for(iL = 0; iL < 10; iL++) {
            for(il = 0; il < 10; il++) {
                if((iTemp1 != iAux2) && (iTemp1 != iAux3) && (iTemp1 != iAux4))
                {
                    iMat[iCont1 * 10 + il][0] = il;
                    iMat[iCont1 * 10 + il][1] = iJ;
                    iMat[iCont1 * 10 + il][2] = iK;
                    iMat[iCont1 * 10 + il][3] = iL;
                } else {
                    iMat[iCont1 * 30 + iTemp2* 10 + il + (int)(0.25 * iAux1)][0]
= il;
                    iMat[iCont1 * 30 + iTemp2* 10 + il + (int)(0.25 * iAux1)][1]
= iJ;
                    iMat[iCont1 * 30 + iTemp2* 10 + il + (int)(0.25 * iAux1)][2]
= iK;
                    iMat[iCont1 * 30 + iTemp2* 10 + il + (int)(0.25 * iAux1)][3]
= iL;
                    iCont2++;

```

```

        };
        iCont3++;
        iCont4++;
    };
    if(iCont3 == 40) {
        iCont1++;
        iCont2 = 0;
        iCont3 = 0;
    };
    if((iCont2 != 30) && (iCont4 < iAux1))
        do
            iTemp2 = random(3);
            while (iMat[iCont1 * 30 + iTemp2 * 10 + (int)(0.25 * iAux1)][0] != -
1);

        if(iTemp1 < 4) iTemp1++;
        else {
            iTemp1 = 1;
            iAux2 = random(4) + 1;
            do
                iAux3 = random(4) + 1;
                while(iAux3 == iAux2);
            do
                iAux4 = random(4) + 1;
                while((iAux4 == iAux2) || (iAux4 == iAux3));
        };
    };

    iCont1 = 0;
    iCont2 = 0;
    for(il = 0; il < (int)(0.25 * iAux1); il++) {
        vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.25 * iAux1), 20);
    };
    for(il = 0; il < (int)(0.75 * iAux1); il++) {
        vGrava_Out_2(usOpcao, iMat[il + (int)(0.25 * iAux1)][0], iMat[il + (int)(0.25 * iAux1)][1],
iMat[il + (int)(0.25 * iAux1)][2], iMat[il + (int)(0.25 * iAux1)][3], 23);
        iCont2++;
        vMostrar_Barra_3(cNome3, iCont2, (int)(0.75 * iAux1), 22);
    };
}

void vTipo_1(const int iAux1, const int iAux2, const unsigned short usOpcao, const char *cNome2, const
char *cNome3) {
    int    iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iCont4 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0,

```

```

        iTemp2 = 0;

vAbrir_Arquivo_Out_1(cNome2);
vAbrir_Arquivo_Out_2(cNome3);
fprintf(fiArqOut1, "%d\n", (int)(0.75 * iAux1));
fprintf(fiArqOut2, "%d\n", (int)(0.25 * iAux1));
for(il = 0; il < iAux1; il++) {
    iMat[iI][0] = -1;
    iMat[iI][1] = -1;
    iMat[iI][2] = -1;
    iMat[iI][3] = -1;
};
iCont1 = 0;
iCont2 = 0;
iCont3 = 0;
iCont4 = 0;
iTemp1 = 1;
iTemp2 = random(3);
for(iJ = 0; iJ < 2; iJ++)
    for(iK = 0; iK < 10; iK++)
        for(iL = 0; iL < 10; iL++) {
            for(il = 0; il < 10; il++) {
                if(iTemp1 != iAux2) {
                    iMat[iCont1 * 30 + iTemp2 * 10 + il][0] = il;
                    iMat[iCont1 * 30 + iTemp2 * 10 + il][1] = iJ;
                    iMat[iCont1 * 30 + iTemp2 * 10 + il][2] = iK;
                    iMat[iCont1 * 30 + iTemp2 * 10 + il][3] = iL;
                    iCont2++;
                } else {
                    iMat[iCont1 * 10 + il + (int)(0.75 * iAux1)][0] = il;
                    iMat[iCont1 * 10 + il + (int)(0.75 * iAux1)][1] = iJ;
                    iMat[iCont1 * 10 + il + (int)(0.75 * iAux1)][2] = iK;
                    iMat[iCont1 * 10 + il + (int)(0.75 * iAux1)][3] = iL;
                };
                iCont3++;
                iCont4++;
            };
            if(iCont3 == 40) {
                iCont1++;
                iCont2 = 0;
                iCont3 = 0;
            };
            if((iCont2 != 30) && (iCont4 < iAux1))
                do
                    iTemp2 = random(3);
                    while (iMat[iCont1 * 30 + iTemp2 * 10][0] != -1);
            if(iTemp1 < 4) iTemp1++;
            else
                iTemp1 = 1;
        };
};
iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.75 * iAux1); il++) {
    vGrava_Out_1(usOpcao, iMat[iI][0], iMat[iI][1], iMat[iI][2], iMat[iI][3], 21);
}

```

```

        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.75 * iAux1), 20);
    };
    for(il = 0; il < (int)(0.25 * iAux1); il++) {
        vGrava_Out_2(usOpcao, iMat[il + (int)(0.75 * iAux1)][0], iMat[il + (int)(0.75 * iAux1)][1],
iMat[il + (int)(0.75 * iAux1)][2], iMat[il + (int)(0.75 * iAux1)][3], 23);
        iCont2++;
        vMostrar_Barra_3(cNome3, iCont2, (int)(0.25 * iAux1), 22);
    };
}

```

```

void vTipo_2(const int iAux1, const int iAux2, const int iAux3, const unsigned short usOpcao, const char
*cNome2, const char *cNome3) {

```

```

    int        iCont1 = 0,
                iCont2 = 0,
                iCont3 = 0,
                iCont4 = 0,
                iCont5 = 0,
                il = 0,
                iJ = 0,
                iK = 0,
                iL = 0,
                iMat[2000][4],
                iTemp1 = 0,
                iTemp2 = 0,
                iTemp3 = 0;

```

```

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.5 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.5 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
        iMat[il][3] = -1;
    };

```

```

    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iCont4 = 0;
    iCont5 = 0;
    iTemp1 = 1;
    iTemp2 = random(2);
    iTemp3 = random(2);
    for(iJ = 0; iJ < 2; iJ++)
        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                for(il = 0; il < 10; il++) {
                    if((iTemp1 != iAux2) && (iTemp1 != iAux3)) {
                        iMat[iCont1 * 20 + iTemp2 * 10 + il][0] = il;
                        iMat[iCont1 * 20 + iTemp2 * 10 + il][1] = iJ;
                        iMat[iCont1 * 20 + iTemp2 * 10 + il][2] = iK;

```

```

        iMat[iCont1 * 20 + iTemp2 * 10 + il][3] = iL;
        iCont2++;
    } else {
        iMat[iCont1 * 20 + iTemp3 * 10 + il + (int)(0.5 * iAux1)][0]
= iI;
        iMat[iCont1 * 20 + iTemp3 * 10 + il + (int)(0.5 * iAux1)][1]
= iJ;
        iMat[iCont1 * 20 + iTemp3 * 10 + il + (int)(0.5 * iAux1)][2]
= iK;
        iMat[iCont1 * 20 + iTemp3 * 10 + il + (int)(0.5 * iAux1)][3]
= iL;
        iCont3++;
    };
    iCont4++;
    iCont5++;
};
if(iCont4 == 40) {
    iCont1++;
    iCont2 = 0;
    iCont3 = 0;
    iCont4 = 0;
};
if((iTemp1 != iAux2) && (iTemp1 != iAux3)) {
    if((iCont2 != 20) && (iCont5 < iAux1))
        do
            iTemp2 = random(2);
            while (iMat[iCont1 * 20 + iTemp2 * 10][0] != -1);
} else {
    if((iCont3 != 20) && (iCont5 < iAux1))
        do
            iTemp3 = random(2);
            while (iMat[iCont1 * 20 + iTemp3 * 10 + (int)(0.5 *
iAux1)][0] != -1);
};
if(iTemp1 < 4) iTemp1++;
else
    iTemp1 = 1;
};

iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
    iCont1++;
    vMostrar_Barra_2(cNome2, iCont1, (int)(0.5 * iAux1), 20);
};
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_2(usOpcao, iMat[il + (int)(0.5 * iAux1)][0], iMat[il + (int)(0.5 * iAux1)][1],
iMat[il + (int)(0.5 * iAux1)][2], iMat[il + (int)(0.5 * iAux1)][3], 23);
    iCont2++;
    vMostrar_Barra_3(cNome3, iCont2, (int)(0.5 * iAux1), 22);
};
}

```

```
void vTipo_3(const int iAux1, const int iAux2, const int iAux3, const int iAux4, const unsigned short
usOpcao, const char *cNome2, const char *cNome3) {
```

```
    int    iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iCont4 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0,
           iTemp2 = 0;

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.25 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.75 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[iJ][0] = -1;
        iMat[iJ][1] = -1;
        iMat[iJ][2] = -1;
        iMat[iJ][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iCont4 = 0;
    iTemp1 = 1;
    iTemp2 = random(3);
    for(iJ = 0; iJ < 2; iJ++)
        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                for(il = 0; il < 10; il++) {
                    if((iTemp1 != iAux2) && (iTemp1 != iAux3) && (iTemp1 != iAux4))

                        iMat[iCont1 * 10 + il][0] = il;
                        iMat[iCont1 * 10 + il][1] = iJ;
                        iMat[iCont1 * 10 + il][2] = iK;
                        iMat[iCont1 * 10 + il][3] = iL;
                    } else {
                        iMat[iCont1 * 30 + iTemp2* 10 + il + (int)(0.25 * iAux1)][0] = il;
                        iMat[iCont1 * 30 + iTemp2* 10 + il + (int)(0.25 * iAux1)][1] = iJ;
                        iMat[iCont1 * 30 + iTemp2* 10 + il + (int)(0.25 * iAux1)][2] = iK;
                        iMat[iCont1 * 30 + iTemp2* 10 + il + (int)(0.25 * iAux1)][3] = iL;
                        iCont2++;
                    };
                    iCont3++;
                    iCont4++;
                };
            };
    if(iCont3 == 40) {
        iCont1++;
        iCont2 = 0;
    }
}
```

```

        iCont3 = 0;
    };
    if((iCont2 != 30) && (iCont4 < iAux1))
        do
            iTemp2 = random(3);
            while (iMat[iCont1 * 30 + iTemp2* 10 + (int)(0.25 * iAux1)][0] != -1);
        if(iTemp1 < 4) iTemp1++;
        else
            iTemp1 = 1;
    };
    iCont1 = 0;
    iCont2 = 0;
    for(il = 0; il < (int)(0.25 * iAux1); il++) {
        vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.25 * iAux1), 20);
    };
    for(il = 0; il < (int)(0.75 * iAux1); il++) {
        vGrava_Out_2(usOpcao, iMat[il + (int)(0.25 * iAux1)][0], iMat[il + (int)(0.25 * iAux1)][1],
iMat[il + (int)(0.25 * iAux1)][2], iMat[il + (int)(0.25 * iAux1)][3], 23);
        iCont2++;
        vMostrar_Barra_3(cNome3, iCont2, (int)(0.75 * iAux1), 22);
    };
}

```

## A.4 Urano

```

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

```

```

FILE    *fiArqIn,
        *fiArqOut1,
        *fiArqOut2;

```

```

void vApresentacao(void);

```

```

void vObtem_Nome_Arquivo(char *cNome1);

```

```

void vObtem_Tipo_Arquivo(unsigned short *usOpcao);

```

```

void vAbrir_Arquivo_In(const char *cNome1);

```

```

void vAbrir_Arquivo_Out_1(const char *cNome2);

```

```

void vAbrir_Arquivo_Out_2(const char *cNome3);

```

```

void vLimpa_Linha(unsigned short int usLinha);

```

```

void vGrava_Out_1(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha);

```



```

void vGrava_Out_2(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const
unsigned short usLinha);

void vMostrar_Barra_1(const char *cNome1, const int iCont1, const int iAux1, const unsigned short
usLinha);

void vMostrar_Barra_2(const char *cNome2, const int iCont2, const int iAux1, const unsigned short
usLinha);

void vMostrar_Barra_3(const char *cNome3, const int iCont3, const int iAux1, const unsigned short
usLinha);

void vTipo_0(const int iAux1, const unsigned short usOpcao);

void vTipo_0_Caso_1(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);

void vTipo_0_Caso_2(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);

void vTipo_0_Caso_3(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3);

void vTipo_1(const int iAux1, const int iAux2, const unsigned short usOpcao, const char *cNome2, const
char *cNome3);

void vTipo_2(const int iAux1, const int iAux2, const int iAux3, const unsigned short usOpcao, const char
*cNome2, const char *cNome3);

void vTipo_3(const int iAux1, const int iAux2, const int iAux3, const int iAux4, const unsigned short
usOpcao, const char *cNome2, const char *cNome3);

void main(void) {
    char    cNome1[81],
            cNome2[81],
            cNome3[81];
    int     iAux1 = 0,
            iAux2 = 0,
            iAux3 = 0,
            iAux4 = 0,
            iCont1 = 0,
            il = 0,
            iTipo1 = 0,
            iTipo2 = 0;
    unsigned short usOpcao;

    randomize();
    vApresentacao();
    vObtem_Nome_Arquivo(cNome1);
    vObtem_Tipo_Arquivo(&usOpcao);
    vAbrir_Arquivo_In(cNome1);
    fscanf(fiArqIn, "%d%d", &iCont1, &iAux1);

```

```

gotoxy(1, 18);
printf("%s 0.0%%", cNome1);
vLimpa_Linha(20);
vLimpa_Linha(21);
vLimpa_Linha(22);
vLimpa_Linha(23);
vTipo_0(iAux1, usOpcao);
fclose(fiArqOut1);
fclose(fiArqOut2);
for(il = 0; il < iCont1; il++) {
    fscanf(fiArqIn, "%d", &iTipo1);
    vLimpa_Linha(20);
    vLimpa_Linha(21);
    vLimpa_Linha(22);
    vLimpa_Linha(23);
    switch(iTipo1) {
        case 0 :
            fscanf(fiArqIn, "%d%s%s", &iTipo2, cNome2, cNome3);
            switch(iTipo2) {
                case 1 :
                    vTipo_0_Caso_1(iAux1, usOpcao, cNome2, cNome3);
                    break;
                case 2 :
                    vTipo_0_Caso_2(iAux1, usOpcao, cNome2, cNome3);
                    break;
                case 3 :
                    vTipo_0_Caso_3(iAux1, usOpcao, cNome2, cNome3);
                    break;
            };
            break;
        case 1 :
            fscanf(fiArqIn, "%d%s%s", &iAux2, cNome2, cNome3);
            vTipo_1(iAux1, iAux2, usOpcao, cNome2, cNome3);
            break;
        case 2 :
            fscanf(fiArqIn, "%d%d%s%s", &iAux2, &iAux3, cNome2, cNome3);
            vTipo_2(iAux1, iAux2, iAux3, usOpcao, cNome2, cNome3);
            break;
        case 3 :
            fscanf(fiArqIn, "%d%d%d%s%s", &iAux2, &iAux3, &iAux4, cNome2, cNome3);
            vTipo_3(iAux1, iAux2, iAux3, iAux4, usOpcao, cNome2, cNome3);
            break;
    };
    fclose(fiArqOut1);
    fclose(fiArqOut2);
    vMostrar_Barra_1(cNome1, il + 1, iCont1, 18);
};
fclose(fiArqIn);
}

void vApresentacao(void) {

    clrscr();

```

```

        printf("Centro Universitario de Brasilia - UniCEUB\n");
        printf("Faculdade de Ciencias Exatas e Tecnologia - FAET\n");
    }

void vObtem_Nome_Arquivo(char *cNome1) {

    printf("Entre com o nome do arquivo gerador de arquivos de treinamento e reconhecimento pela
RNA:\nNome = ");
    scanf("%s", cNome1);
}

void vObtem_Tipo_Arquivo(unsigned short *usOpcao) {

    printf("\nEscolha qual o tipo de figuras sera manipulado:\n1 - Digitos\n2 - Vogais\nOpcao = ");
    scanf("%u", usOpcao);
    if((*usOpcao != 1) && (*usOpcao != 2))
        *usOpcao = 1;
}

void vAbrir_Arquivo_In(const char *cNome1) {

    fiArqIn = fopen(cNome1, "rt");
    if(fiArqIn == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome1);
        exit(1);
    };
}

void vAbrir_Arquivo_Out_1(const char *cNome2) {

    fiArqOut1 = fopen(cNome2, "wt");
    if(fiArqOut1 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome2);
        exit(1);
    };
}

void vAbrir_Arquivo_Out_2(const char *cNome3) {

    fiArqOut2 = fopen(cNome3, "wt");
    if(fiArqOut2 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome3);
        exit(1);
    };
}

void vLimpa_Linha(unsigned short int usLinha) {

    unsigned short usl;

    for(ysl = 0; ysl < 80; ysl++) {
        gotoxy(ysl, usLinha);
        printf(" ");
    };
}

```

```
}
```

```
void vGrava_Out_1(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const unsigned short usLinha) {
```

```
    if(usOpcao == 1) {
        fprintf(fiArqOut1, "%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    } else {
        fprintf(fiArqOut1, "%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    };
}
```

```
void vGrava_Out_2(const unsigned short usOpcao, const int il, const int iJ, const int iK, const int iL, const unsigned short usLinha) {
```

```
    if(usOpcao == 1) {
        fprintf(fiArqOut2, "%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d DIG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    } else {
        fprintf(fiArqOut2, "%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
        gotoxy(1, usLinha);
        printf("%d VOG%d_%d%d%d.IMG\n", il, il, iJ, iK, iL);
    };
}
```

```
void vMostrar_Barra_1(const char *cNome1, const int iCont1, const int iAux1, const unsigned short usLinha) {
```

```
    int iM;
    float fPorc;

    gotoxy(1, usLinha);
    printf("%s ", cNome1);
    fPorc = (1.0 * iCont1) / iAux1;
    for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
        gotoxy(strlen(cNome1) + 2 + iM, usLinha);
        printf("*");
    };
    gotoxy(strlen(cNome1) + 2 + iM, usLinha);
    printf(" %.1f %%", 100.0*fPorc);
}
```

```
void vMostrar_Barra_2(const char *cNome2, const int iCont2, const int iAux1, const unsigned short usLinha) {
```

```
    int iM;
    float fPorc;

    gotoxy(1, usLinha);
```

```

printf("%s ", cNome2);
fPorc = (1.0 * iCont2) / iAux1;
for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
    gotoxy(strlen(cNome2) + 2 + iM, usLinha);
    printf("**");
};
gotoxy(strlen(cNome2) + 2 + iM, usLinha);
printf(" %.1f %%%", 100.0*fPorc);
}

```

```

void vMostrar_Barra_3(const char *cNome3, const int iCont3, const int iAux1, const unsigned short
usLinha) {

```

```

    int iM;
    float fPorc;

    gotoxy(1, usLinha);
    printf("%s ", cNome3);
    fPorc = (1.0 * iCont3) / iAux1;
    for(iM = 0; iM <= (int)(50.0 * fPorc); iM++) {
        gotoxy(strlen(cNome3) + 2 + iM, usLinha);
        printf("**");
    };
    gotoxy(strlen(cNome3) + 2 + iM, usLinha);
    printf(" %.1f %%%", 100.0*fPorc);
}

```

```

void vTipo_0(const int iAux1, const unsigned short usOpcao) {

```

```

    int    iCont1 = 0,
           iCont2 = 0,
           iI = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4];

    if(usOpcao == 1) {
        vAbrir_Arquivo_Out_1("DUrat00.txt");
        vAbrir_Arquivo_Out_2("DUrar00.txt");
    } else {
        vAbrir_Arquivo_Out_1("VUrat00.txt");
        vAbrir_Arquivo_Out_2("VUrar00.txt");
    };
    fprintf(fiArqOut1, "%d\n", iAux1);
    fprintf(fiArqOut2, "%d\n", iAux1);
    for(iI = 0; iI < iAux1; iI++) {
        iMat[iI][0] = -1;
        iMat[iI][1] = -1;
        iMat[iI][2] = -1;
        iMat[iI][3] = -1;
    };
    iCont1 = 0;
    for(iJ = 0; iJ < 2; iJ++)

```

```

        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                for(il = 0; il < 10; il++) {
                    iMat[iCont1 * 10 + il][0] = il;
                    iMat[iCont1 * 10 + il][1] = iJ;
                    iMat[iCont1 * 10 + il][2] = iK;
                    iMat[iCont1 * 10 + il][3] = iL;
                };
                iCont1++;
            };
iCont1 = 0;
iCont2 = 0;
for(il = 0; il < iAux1; il++) {
    vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
    iCont1++;
    if(usOpcao == 1)
        vMostrar_Barra_2("DUrat00.txt", iCont1, iAux1, 20);
    else
        vMostrar_Barra_2("VUrat00.txt", iCont1, iAux1, 20);
};
for(il = 0; il < iAux1; il++) {
    vGrava_Out_2(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 23);
    iCont2++;
    if(usOpcao == 1)
        vMostrar_Barra_3("DUrar00.txt", iCont2, iAux1, 22);
    else
        vMostrar_Barra_3("VUrar00.txt", iCont2, iAux1, 22);
};
}

```

```

void vTipo_0_Caso_1(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {

```

```

    int    iAux2 = 0,
           iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iCont4 = 0,
           iCont5 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0;

```

```

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.75 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.25 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
    }

```

```

        iMat[iI][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iCont4 = 0;
    iCont5 = 0;
    iTemp1 = 1;
    iAux2 = random(4)+1;
    for(iJ = 0; iJ < 2; iJ++)
        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                for(iI = 0; iI < 10; iI++) {
                    if(iTemp1 != iAux2) {
                        iMat[iCont1 * 30 + iCont3 * 10 + iI][0] = iI;
                        iMat[iCont1 * 30 + iCont3 * 10 + iI][1] = iJ;
                        iMat[iCont1 * 30 + iCont3 * 10 + iI][2] = iK;
                        iMat[iCont1 * 30 + iCont3 * 10 + iI][3] = iL;
                        iCont2++;
                    } else {
                        iMat[iCont1 * 10 + iI + (int)(0.75 * iAux1)][0] = iI;
                        iMat[iCont1 * 10 + iI + (int)(0.75 * iAux1)][1] = iJ;
                        iMat[iCont1 * 10 + iI + (int)(0.75 * iAux1)][2] = iK;
                        iMat[iCont1 * 10 + iI + (int)(0.75 * iAux1)][3] = iL;
                        iCont4++;
                    }
                };
            };
        if(iTemp1 != iAux2)
            iCont3++;
        iCont5++;
        if(iCont5 == 4) {
            iCont1++;
            iCont2 = 0;
            iCont3 = 0;
            iCont4 = 0;
            iCont5 = 0;
        };
        if(iTemp1 < 4) iTemp1++;
        else {
            iTemp1 = 1;
            iAux2 = random(4)+1;
        };
    };

    iCont1 = 0;
    iCont2 = 0;
    for(iI = 0; iI < (int)(0.75 * iAux1); iI++) {
        vGrava_Out_1(usOpcao, iMat[iI][0], iMat[iI][1], iMat[iI][2], iMat[iI][3], 21);
        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.75 * iAux1), 20);
    };
    for(iI = 0; iI < (int)(0.25 * iAux1); iI++) {
        vGrava_Out_2(usOpcao, iMat[iI + (int)(0.75 * iAux1)][0], iMat[iI + (int)(0.75 * iAux1)][1],
        iMat[iI + (int)(0.75 * iAux1)][2], iMat[iI + (int)(0.75 * iAux1)][3], 23);
        iCont2++;
    };

```

```

        vMostrar_Barra_3(cNome3, iCont2, (int)(0.25 * iAux1), 22);
    };
}

void vTipo_0_Caso_2(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {

    int    iAux2 = 0,
           iAux3 = 0,
           iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iCont4 = 0,
           iCont5 = 0,
           iCont6 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0;

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.5 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.5 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
        iMat[il][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iCont4 = 0;
    iCont5 = 0;
    iCont6 = 0;
    iTemp1 = 1;
    iAux2 = random(4)+1;
    do
        iAux3 = random(4) + 1;
    while(iAux3 == iAux2);
    for(iJ = 0; iJ < 2; iJ++)
        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                for(il = 0; il < 10; il++) {
                    if((iTemp1 != iAux2) && (iTemp1 != iAux3)) {
                        iMat[iCont1 * 20 + iCont3 * 10 + il][0] = il;
                        iMat[iCont1 * 20 + iCont3 * 10 + il][1] = iJ;
                        iMat[iCont1 * 20 + iCont3 * 10 + il][2] = iK;
                        iMat[iCont1 * 20 + iCont3 * 10 + il][3] = iL;
                        iCont2++;
                    } else {

```



```

        iMat[iCont1 * 20 + iCont5 * 10 + il + (int)(0.5 * iAux1)][0] = il;
        iMat[iCont1 * 20 + iCont5 * 10 + il + (int)(0.5 * iAux1)][1] = iJ;
        iMat[iCont1 * 20 + iCont5 * 10 + il + (int)(0.5 * iAux1)][2] = iK;
        iMat[iCont1 * 20 + iCont5 * 10 + il + (int)(0.5 * iAux1)][3] = iL;
        iCont4++;
    };
};
if((iTemp1 != iAux2) && (iTemp1 != iAux3))
    iCont3++;
else
    iCont5++;
iCont6++;
if(iCont6 == 4) {
    iCont1++;
    iCont2 = 0;
    iCont3 = 0;
    iCont4 = 0;
    iCont5 = 0;
};
if(iTemp1 < 4) iTemp1++;
else {
    iTemp1 = 1;
    iAux2 = random(4) + 1;
    do
        iAux3 = random(4) + 1;
    while(iAux3 == iAux2);
};
};

iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
    iCont1++;
    vMostrar_Barra_2(cNome2, iCont1, (int)(0.5 * iAux1), 20);
};
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_2(usOpcao, iMat[il + (int)(0.5 * iAux1)][0], iMat[il + (int)(0.5 * iAux1)][1],
iMat[il + (int)(0.5 * iAux1)][2], iMat[il + (int)(0.5 * iAux1)][3], 23);
    iCont2++;
    vMostrar_Barra_3(cNome3, iCont2, (int)(0.5 * iAux1), 22);
};
}

void vTipo_0_Caso_3(const int iAux1, const unsigned short usOpcao, const char *cNome2, const char
*cNome3) {
    int    iAux2 = 0,
           iAux3 = 0,
           iAux4 = 0,
           iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iCont4 = 0,
           iCont5 = 0,

```

```

        il = 0,
        iJ = 0,
        iK = 0,
        iL = 0,
        iMat[2000][4],
        iTemp1 = 0;

vAbrir_Arquivo_Out_1(cNome2);
vAbrir_Arquivo_Out_2(cNome3);
fprintf(fiArqOut1, "%d\n", (int)(0.25 * iAux1));
fprintf(fiArqOut2, "%d\n", (int)(0.75 * iAux1));
for(il = 0; il < iAux1; il++) {
    iMat[iJ][0] = -1;
    iMat[iJ][1] = -1;
    iMat[iJ][2] = -1;
    iMat[iJ][3] = -1;
};
iCont1 = 0;
iCont2 = 0;
iCont3 = 0;
iCont4 = 0;
iCont5 = 0;
iTemp1 = 1;
iAux2 = random(4)+1;
do
    iAux3 = random(4) + 1;
while(iAux3 == iAux2);
do
    iAux4 = random(4) + 1;
while((iAux4 == iAux2) || (iAux4 == iAux3));
for(iJ = 0; iJ < 2; iJ++)
    for(iK = 0; iK < 10; iK++)
        for(iL = 0; iL < 10; iL++) {
            for(il = 0; il < 10; il++) {
                if((iTemp1 != iAux2) && (iTemp1 != iAux3) && (iTemp1 != iAux4))
                {
                    iMat[iCont1 * 10 + il][0] = il;
                    iMat[iCont1 * 10 + il][1] = iJ;
                    iMat[iCont1 * 10 + il][2] = iK;
                    iMat[iCont1 * 10 + il][3] = iL;
                    iCont2++;
                } else {
                    iMat[iCont1 * 30 + iCont4 * 10 + il + (int)(0.25 * iAux1)][0] = il;
                    iMat[iCont1 * 30 + iCont4 * 10 + il + (int)(0.25 * iAux1)][1] = iJ;
                    iMat[iCont1 * 30 + iCont4 * 10 + il + (int)(0.25 * iAux1)][2] = iK;
                    iMat[iCont1 * 30 + iCont4 * 10 + il + (int)(0.25 * iAux1)][3] = iL;
                    iCont3++;
                }
            };
        };
if(!((iTemp1 != iAux2) && (iTemp1 != iAux3) && (iTemp1 != iAux4)))
    iCont4++;
iCont5++;
if(iCont5 == 4) {
    iCont1++;

```

```

        iCont2 = 0;
        iCont3 = 0;
        iCont4 = 0;
        iCont5 = 0;
    };
    if(iTemp1 < 4) iTemp1++;
    else {
        iTemp1 = 1;
        iAux2 = random(4) + 1;
        do
            iAux3 = random(4) + 1;
        while(iAux3 == iAux2);
        do
            iAux4 = random(4) + 1;
        while((iAux4 == iAux2) || (iAux4 == iAux3));
    };
};

iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.25 * iAux1); il++) {
    vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
    iCont1++;
    vMostrar_Barra_2(cNome2, iCont1, (int)(0.25 * iAux1), 20);
};
for(il = 0; il < (int)(0.75 * iAux1); il++) {
    vGrava_Out_2(usOpcao, iMat[il + (int)(0.25 * iAux1)][0], iMat[il + (int)(0.25 * iAux1)][1],
iMat[il + (int)(0.25 * iAux1)][2], iMat[il + (int)(0.25 * iAux1)][3], 23);
    iCont2++;
    vMostrar_Barra_3(cNome3, iCont2, (int)(0.75 * iAux1), 22);
};
}

```

```

void vTipo_1(const int iAux1, const int iAux2, const unsigned short usOpcao, const char *cNome2, const
char *cNome3) {

```

```

    int    iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iCont4 = 0,
           iCont5 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0;

```

```

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.75 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.25 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
    }

```

```

        iMat[iI][2] = -1;
        iMat[iI][3] = -1;
    };
    iCont1 = 0;
    iCont2 = 0;
    iCont3 = 0;
    iCont4 = 0;
    iCont5 = 0;
    iTemp1 = 1;
    for(iJ = 0; iJ < 2; iJ++)
        for(iK = 0; iK < 10; iK++)
            for(iL = 0; iL < 10; iL++) {
                for(iI = 0; iI < 10; iI++) {
                    if(iTemp1 != iAux2) {
                        iMat[iCont1 * 30 + iCont3 * 10 + iI][0] = iI;
                        iMat[iCont1 * 30 + iCont3 * 10 + iI][1] = iJ;
                        iMat[iCont1 * 30 + iCont3 * 10 + iI][2] = iK;
                        iMat[iCont1 * 30 + iCont3 * 10 + iI][3] = iL;
                        iCont2++;
                    } else {
                        iMat[iCont1 * 10 + iI + (int)(0.75 * iAux1)][0] = iI;
                        iMat[iCont1 * 10 + iI + (int)(0.75 * iAux1)][1] = iJ;
                        iMat[iCont1 * 10 + iI + (int)(0.75 * iAux1)][2] = iK;
                        iMat[iCont1 * 10 + iI + (int)(0.75 * iAux1)][3] = iL;
                        iCont4++;
                    }
                };
            };
        if(iTemp1 != iAux2)
            iCont3++;
        iCont5++;
        if(iCont5 == 4) {
            iCont1++;
            iCont2 = 0;
            iCont3 = 0;
            iCont4 = 0;
            iCont5 = 0;
        };
        if(iTemp1 < 4) iTemp1++;
        else
            iTemp1 = 1;
    };

    iCont1 = 0;
    iCont2 = 0;
    for(iI = 0; iI < (int)(0.75 * iAux1); iI++) {
        vGrava_Out_1(usOpcao, iMat[iI][0], iMat[iI][1], iMat[iI][2], iMat[iI][3], 21);
        iCont1++;
        vMostrar_Barra_2(cNome2, iCont1, (int)(0.75 * iAux1), 20);
    };
    for(iI = 0; iI < (int)(0.25 * iAux1); iI++) {
        vGrava_Out_2(usOpcao, iMat[iI + (int)(0.75 * iAux1)][0], iMat[iI + (int)(0.75 * iAux1)][1],
iMat[iI + (int)(0.75 * iAux1)][2], iMat[iI + (int)(0.75 * iAux1)][3], 23);
        iCont2++;
        vMostrar_Barra_3(cNome3, iCont2, (int)(0.25 * iAux1), 22);
    };
};

```

```
}
```

```
void vTipo_2(const int iAux1, const int iAux2, const int iAux3, const unsigned short usOpcao, const char  
*cNome2, const char *cNome3) {
```

```
    int      iCont1 = 0,  
             iCont2 = 0,  
             iCont3 = 0,  
             iCont4 = 0,  
             iCont5 = 0,  
             iCont6 = 0,  
             il = 0,  
             iJ = 0,  
             iK = 0,  
             iL = 0,  
             iMat[2000][4],  
             iTemp1 = 0;
```

```
    vAbrir_Arquivo_Out_1(cNome2);  
    vAbrir_Arquivo_Out_2(cNome3);  
    fprintf(fiArqOut1, "%d\n", (int)(0.5 * iAux1));  
    fprintf(fiArqOut2, "%d\n", (int)(0.5 * iAux1));  
    for(il = 0; il < iAux1; il++) {
```

```
        iMat[iJ][0] = -1;  
        iMat[iJ][1] = -1;  
        iMat[iJ][2] = -1;  
        iMat[iJ][3] = -1;
```

```
    };
```

```
    iCont1 = 0;
```

```
    iCont2 = 0;
```

```
    iCont3 = 0;
```

```
    iCont4 = 0;
```

```
    iCont5 = 0;
```

```
    iCont6 = 0;
```

```
    iTemp1 = 1;
```

```
    for(iJ = 0; iJ < 2; iJ++)
```

```
        for(iK = 0; iK < 10; iK++)
```

```
            for(iL = 0; iL < 10; iL++) {
```

```
                for(il = 0; il < 10; il++) {
```

```
                    if((iTemp1 != iAux2) && (iTemp1 != iAux3)) {
```

```
                        iMat[iCont1 * 20 + iCont3 * 10 + il][0] = il;
```

```
                        iMat[iCont1 * 20 + iCont3 * 10 + il][1] = iJ;
```

```
                        iMat[iCont1 * 20 + iCont3 * 10 + il][2] = iK;
```

```
                        iMat[iCont1 * 20 + iCont3 * 10 + il][3] = iL;
```

```
                        iCont2++;
```

```
                    } else {
```

```
                        iMat[iCont1 * 20 + iCont5 * 10 + il + (int)(0.5 * iAux1)][0] = il;
```

```
                        iMat[iCont1 * 20 + iCont5 * 10 + il + (int)(0.5 * iAux1)][1] = iJ;
```

```
                        iMat[iCont1 * 20 + iCont5 * 10 + il + (int)(0.5 * iAux1)][2] = iK;
```

```
                        iMat[iCont1 * 20 + iCont5 * 10 + il + (int)(0.5 * iAux1)][3] = iL;
```

```
                        iCont4++;
```

```
                    };
```

```
            };
```

```
        if((iTemp1 != iAux2) && (iTemp1 != iAux3))
```

```

        iCont3++;
    else
        iCont5++;
    iCont6++;
    if(iCont6 == 4) {
        iCont1++;
        iCont2 = 0;
        iCont3 = 0;
        iCont4 = 0;
        iCont5 = 0;
    };
    if(iTemp1 < 4) iTemp1++;
    else
        iTemp1 = 1;
};

iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
    iCont1++;
    vMostrar_Barra_2(cNome2, iCont1, (int)(0.5 * iAux1), 20);
};
for(il = 0; il < (int)(0.5 * iAux1); il++) {
    vGrava_Out_2(usOpcao, iMat[il + (int)(0.5 * iAux1)][0], iMat[il + (int)(0.5 * iAux1)][1],
iMat[il + (int)(0.5 * iAux1)][2], iMat[il + (int)(0.5 * iAux1)][3], 23);
    iCont2++;
    vMostrar_Barra_3(cNome3, iCont2, (int)(0.5 * iAux1), 22);
};
}

```

```

void vTipo_3(const int iAux1, const int iAux2, const int iAux3, const int iAux4, const unsigned short
usOpcao, const char *cNome2, const char *cNome3) {

```

```

    int    iCont1 = 0,
           iCont2 = 0,
           iCont3 = 0,
           iCont4 = 0,
           iCont5 = 0,
           il = 0,
           iJ = 0,
           iK = 0,
           iL = 0,
           iMat[2000][4],
           iTemp1 = 0;

```

```

    vAbrir_Arquivo_Out_1(cNome2);
    vAbrir_Arquivo_Out_2(cNome3);
    fprintf(fiArqOut1, "%d\n", (int)(0.25 * iAux1));
    fprintf(fiArqOut2, "%d\n", (int)(0.75 * iAux1));
    for(il = 0; il < iAux1; il++) {
        iMat[il][0] = -1;
        iMat[il][1] = -1;
        iMat[il][2] = -1;
        iMat[il][3] = -1;
    }

```

```

};
iCont1 = 0;
iCont2 = 0;
iCont3 = 0;
iCont4 = 0;
iCont5 = 0;
iTemp1 = 1;
for(iJ = 0; iJ < 2; iJ++)
    for(iK = 0; iK < 10; iK++)
        for(iL = 0; iL < 10; iL++) {
            for(il = 0; il < 10; il++) {
                if((iTemp1 != iAux2) && (iTemp1 != iAux3) && (iTemp1 != iAux4))
{
                    iMat[iCont1 * 10 + il][0] = il;
                    iMat[iCont1 * 10 + il][1] = iJ;
                    iMat[iCont1 * 10 + il][2] = iK;
                    iMat[iCont1 * 10 + il][3] = iL;
                    iCont2++;
                } else {
                    iMat[iCont1 * 30 + iCont4 * 10 + il + (int)(0.25 * iAux1)][0] = il;
                    iMat[iCont1 * 30 + iCont4 * 10 + il + (int)(0.25 * iAux1)][1] = iJ;
                    iMat[iCont1 * 30 + iCont4 * 10 + il + (int)(0.25 * iAux1)][2] = iK;
                    iMat[iCont1 * 30 + iCont4 * 10 + il + (int)(0.25 * iAux1)][3] = iL;
                    iCont3++;
                }
            };
        };
        if(!((iTemp1 != iAux2) && (iTemp1 != iAux3) && (iTemp1 != iAux4)))
            iCont4++;
        iCont5++;
        if(iCont5 == 4) {
            iCont1++;
            iCont2 = 0;
            iCont3 = 0;
            iCont4 = 0;
            iCont5 = 0;
        };
        if(iTemp1 < 4) iTemp1++;
        else
            iTemp1 = 1;
    };
iCont1 = 0;
iCont2 = 0;
for(il = 0; il < (int)(0.25 * iAux1); il++) {
    vGrava_Out_1(usOpcao, iMat[il][0], iMat[il][1], iMat[il][2], iMat[il][3], 21);
    iCont1++;
    vMostrar_Barra_2(cNome2, iCont1, (int)(0.25 * iAux1), 20);
};
for(il = 0; il < (int)(0.75 * iAux1); il++) {
    vGrava_Out_2(usOpcao, iMat[il + (int)(0.25 * iAux1)][0], iMat[il + (int)(0.25 * iAux1)][1],
iMat[il + (int)(0.25 * iAux1)][2], iMat[il + (int)(0.25 * iAux1)][3], 23);
    iCont2++;
    vMostrar_Barra_3(cNome3, iCont2, (int)(0.75 * iAux1), 22);
};
}

```

## A.5 Saturno0

```
#include <alloc.h>
#include <conio.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

char    cNome1[81],
        cNome2[81],
        cNome3[81],
        cNome4[81],
        cNome5[81],
        cNome6[81],
        cNome7[81],
        cNome8[81];

float    huge    *pdPeso[10],
        huge    *pdImagem;
FILE     *fiArqIn1,
        *fiArqIn2,
        *fiArqIn3,
        *fiArqIn4,
        *fiArqIn5,
        *fiArqIn6,
        *fiArqOut1,
        *fiArqOut2;

void vApresentacao(void);

void vOtem_Tipo_de_Dado(unsigned short *usOpcao_Dado);

void vObtem_Nome_In_1(char *cNome1);

void vLer_Nome_In_2(char *cNome2);

void vLer_Nome_In_3(char *cNome3);

void vLer_Nome_In_4(char *cNome4);

void vLer_Nome_In_5(int *iValor, char *cNome5);

void vLer_Nome_In_6(int *iValor, char *cNome6);

void vLer_Nome_Out1(char *cNome7);

void vLer_Nome_Out2(char *cNome8);
```



```

void vAbrir_Arquivo_In_1(const char *cNome1);

void vAbrir_Arquivo_In_2(const char *cNome2);

void vAbrir_Arquivo_In_3(const char *cNome3);

void vCria_Espaco(const int *iNNCam, unsigned short *usOpcao_Inic);

void vLer_Peso(const int *iNNCam);

void vGravar_Peso(const int iNCam, const int *iNNCam);

void vGrava_Cabecalho_Html(const unsigned short usOpcao_Dado, const unsigned short
usOpcao_Modo);

void vGrava_Informacoes_Html(const unsigned short usOpcao_Dado, const unsigned short
usOpcao_Modo, const float fEta, const int iEpocas, const int iCont2, const int iCont3, const char
*cNome1, const char *cNome2, const char *cNome3, const char *cNome4, const char *cNome7, const
char *cNome8);

void vGrava_Informacoes_Pesos_Html(const int iNCam, const int *iNNCam, const char *cNome4);

void vAnaliza_Conjunto_1(const int *iNNCam, const int iCont2, const unsigned short usOpcao_Dado);

void vAnaliza_Conjunto_2(const int *iNNCam, const int iCont3, const unsigned short usOpcao_Dado);

void vTreina_RNA(const int *iNNCam, const float fEta, const int iCont, const int iInter);

void vReconhe_RNA_T(const int *iNNCam, const int iEpocas, const unsigned short usOpcao_Dado);

void vReconhe_RNA_R(const int *iNNCam, const int iEpocas, const unsigned short usOpcao_Dado);

void vGrava_Nome_RNA_T(const int *iNNCam, const unsigned short usOpcao_Dado);

void vGrava_Nome_RNA_R(const int *iNNCam, const unsigned short usOpcao_Dado);

void vLer_Imagem(const int *iNNCam);

float dSigmoid(const float dSoma);

int iMaximo(const float *dVetorResposta);

void main(void) {
    int    iNCam = 0,
           iNNCam[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
           iCont2 = 0,
           iCont3 = 0,
           iI = 0,
           iJ = 0,
           iK = 0,
           iEpocas = 0;
    float  fEta = 0;
    time_t tLTInicio,

```

```

        tLTFim;
unsigned short usCont = 0,

        usOpcao_Inic = 0,
        usOpcao_Dado = 0,
        usOpcao_Modo = 1;

vApresentacao();
vObtem_Nome_In_1(cNome1);
vAbrir_Arquivo_In_1(cNome1);
fscanf(fiArqIn1, "%u%f%d", &usCont, &fEta, &iEpocas);
for(il = 0; il < usCont; il++) {
    tLTInicio = time(NULL);
    fscanf(fiArqIn1, "%u%u", &usOpcao_Modo, &usOpcao_Dado);
    vLer_Nome_In_2(cNome2);
    vLer_Nome_In_3(cNome3);
    vLer_Nome_In_4(cNome4);
    vLer_Nome_Out1(cNome7);
    vLer_Nome_Out2(cNome8);
    clrscr();
    vApresentacao();
    gotoxy(1,9);
    if(usOpcao_Dado == 1)
        printf("Tipo de Dado: Digitos;");
    else
        printf("Tipo de Dado: Vogais;");
    gotoxy(1,10);
    printf("Contador = %3d;", il + 1);
    iJ = 0;
    gotoxy(1,11);
    printf("Epoca = %6d;", iJ);
    fread(&iNCam, sizeof(iNCam), 1, fiArqIn4);
    for(iJ = 0; iJ < iNCam; iJ++)
        fread(&iNNCam[iJ], sizeof(iNNCam[iJ]), 1, fiArqIn4);
    vCria_Espaco(iNNCam, &usOpcao_Inic);
    vLer_Peso(iNNCam);
    vGrava_Cabecalho_Html(usOpcao_Dado, usOpcao_Modo);
    fscanf(fiArqIn2, "%d", &iCont2);
    fscanf(fiArqIn3, "%d", &iCont3);
    vGrava_Informacoes_Html(usOpcao_Dado, usOpcao_Modo, fEta, iEpocas, iCont2,
iCont3, cNome1, cNome2, cNome3, cNome4, cNome7, cNome8);
    vGrava_Informacoes_Pesos_Html(iNCam, iNNCam, cNome4);
    vAnaliza_Conjunto_1(iNNCam, iCont2, usOpcao_Dado);
    vAnaliza_Conjunto_2(iNNCam, iCont3, usOpcao_Dado);
    iK = (int)(iEpocas * 0.005);
    for(iJ = 1; iJ <= iEpocas; iJ++) {
        gotoxy(1,10);
        printf("Contador = %3d;", il + 1);
        gotoxy(1,11);
        printf("Epoca = %6d;", iJ);
        rewind(fiArqIn2);
        vTreina_RNA(iNNCam, fEta, iJ, iK);
        if((iJ == (iEpocas / 4)) || (iJ == (iEpocas / 2)) || (iJ == (3 * iEpocas / 4)) || (iJ ==
iEpocas)) {
            rewind(fiArqIn2);

```

```

        rewind(fiArqIn3);
        vReconhe_RNA_T(iNNCam, iJ, usOpcao_Dado);
        vReconhe_RNA_R(iNNCam, iJ, usOpcao_Dado);
    };
};
vGravar_Peso(iNCam, iNNCam);
rewind(fiArqIn2);
rewind(fiArqIn3);
vGrava_Nome_RNA_T(iNNCam, usOpcao_Dado);
vGrava_Nome_RNA_R(iNNCam, usOpcao_Dado);
tLTFim = time(NULL);
fprintf(fiArqOut1, "<p>\n<table align = \"center\" border = 1>\n");
fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Data e Hora
Inicial</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", ctime(&tLTInicio));
fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Data e Hora
Inicial</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", ctime(&tLTFim));
fprintf(fiArqOut1, "</table>\n");
fclose(fiArqIn2);
fclose(fiArqIn3);
fclose(fiArqIn4);
fprintf(fiArqOut1, "</body>\n</html>");
fclose(fiArqOut1);
fclose(fiArqOut2);
}
fclose(fiArqIn1);
farfree(pdImagem);
farfree(pdPeso);
}

void vApresentacao(void) {

    clrscr();
    printf("Centro Universitario de Brasilia - UniCEUB\n");
    printf("Faculdade de Ciencias Exatas e Tecnologia - FAET\n");
}

void vOtem_Tipo_de_Dado(unsigned short *usOpcao_Dado) {

    printf("Entre com a opcao do tipo de dado a ser usado:\n1 = Digito.\n2 = Vogal.\nOpcao Dado =
");
    scanf("%u", usOpcao_Dado);
    if((*usOpcao_Dado != 1) && (*usOpcao_Dado != 2))
        *usOpcao_Dado = 1;
}

void vObtem_Tipo_de_Modo(unsigned short *usOpcao_Modo) {

    printf("\nEntre com a opcao do tipo do modo de treinamento e reconhecimento a ser usado:\n1 =
Sedna.\n2 = Plutao.\nOpcao Modo = ");
    scanf("%u", usOpcao_Modo);
    if((*usOpcao_Modo != 1) && (*usOpcao_Modo != 2))
        *usOpcao_Modo = 1;
}

```

```

void vObtem_Nome_In_1(char *cNome1) {

    printf("\nEntre com o nome do arquivo completo de treinamento e reconhecimento da
RNA:\nNome = ");
    scanf("%s", cNome1);
}

void vLer_Nome_In_2(char *cNome2) {

    fscanf(fiArqIn1, "%s", cNome2);
    fiArqIn2 = fopen(cNome2, "rt+");
    if(fiArqIn2 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome2);
        exit(1);
    };
}

void vLer_Nome_In_3(char *cNome3) {

    fscanf(fiArqIn1, "%s", cNome3);
    fiArqIn3 = fopen(cNome3, "rt+");
    if(fiArqIn3 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome3);
        exit(1);
    };
}

void vLer_Nome_In_4(char *cNome4) {

    fscanf(fiArqIn1, "%s", cNome4);
    fiArqIn4 = fopen(cNome4, "rb+");
    if(fiArqIn4 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome4);
        exit(1);
    };
}

void vLer_Nome_In_5(int *iValor, char *cNome5) {

    fscanf(fiArqIn2, "%d%s", iValor, cNome5);
    fiArqIn5 = fopen(cNome5, "rb");
    if(fiArqIn5 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome5);
        exit(1);
    };
}

void vLer_Nome_In_6(int *iValor, char *cNome6) {

    fscanf(fiArqIn3, "%d%s", iValor, cNome6);
    fiArqIn6 = fopen(cNome6, "rb");
    if(fiArqIn6 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome6);
        exit(1);
    };
}

```

```

    };
}

void vLer_Nome_Out1(char *cNome7) {

    fscanf(fiArqIn1,"%s",cNome7);
    fiArqOut1 = fopen(cNome7, "wt");
    if(fiArqOut1 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome7);
        exit(1);
    };
}

void vLer_Nome_Out2(char *cNome8) {

    fscanf(fiArqIn1,"%s",cNome8);
    fiArqOut2 = fopen(cNome8, "wt");
    if(fiArqOut2 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome8);
        exit(1);
    };
}

void vAbrir_Arquivo_In_1(const char *cNome1) {

    fiArqIn1 = fopen(cNome1, "rt");
    if(fiArqIn1 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome1);
        exit(1);
    };
}

void vAbrir_Arquivo_In_2(const char *cNome2) {

    fiArqIn2 = fopen(cNome2, "rt");
    if(fiArqIn2 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome2);
        exit(1);
    };
}

void vAbrir_Arquivo_In_3(const char *cNome3) {

    fiArqIn3 = fopen(cNome3, "rt");
    if(fiArqIn3 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome3);
        exit(1);
    };
}

void vCria_Espaco(const int *iNNCam, unsigned short *usOpcao_Inic) {

    int il;

```

```

        if(*usOpcao_Inic == 0) {
            pdImagem = (float huge *) farmalloc((iNNCam[0] + 1) * sizeof(float));
            for(il = 0; il < 10; il++)
                pdPeso[il] = (float huge *) farmalloc((iNNCam[0] + 1) * sizeof(float)) ;
            *usOpcao_Inic = 1;
        };
    }

void vLer_Peso(const int *iNNCam) {

    int    il = 0,
           iJ = 0;

    for(il = 0; il < iNNCam[1]; il++)
        for(iJ = 0; iJ < (iNNCam[0] + 1); iJ++)
            fread(&pdPeso[il][iJ], sizeof(pdPeso[il][iJ]), 1, fiArqIn4);
}

void vGravar_Peso(const int iNCam, const int *iNNCam) {

    int    il = 0,
           iJ = 0;

    rewind(fiArqIn4);
    fwrite(&iNCam, sizeof(iNCam), 1, fiArqIn4);
    for(il = 0; il < iNCam; il++)
        fwrite(&iNNCam[il], sizeof(iNNCam[il]), 1, fiArqIn4);
    for(il = 0; il < iNNCam[1]; il++)
        for(iJ = 0; iJ < (iNNCam[0] + 1); iJ++)
            fwrite(&pdPeso[il][iJ], sizeof(pdPeso[il][iJ]), 1, fiArqIn4);
}

void vGrava_Cabecalho_Html(const unsigned short usOpcao_Dado, const unsigned short
usOpcao_Modo) {

    fprintf(fiArqOut1, "<html>\n<head>\n");
    fprintf(fiArqOut1, "<title>Treinamento e Reconhecimento de RNA com 0 camadas
ocultas</title>\n");
    fprintf(fiArqOut1, "<\head><body bgcolor = \"white\" text = \"black\">\n");
    fprintf(fiArqOut1, "<h1 align = \"center\"><b>Resultado do Treinamento e Reconhecimento de ");
    if(usOpcao_Dado == 1)
        fprintf(fiArqOut1, "Dígitos pelas Redes Neurais Artificiais</b></h1>\n");
    else
        fprintf(fiArqOut1, "Vogais pelas Redes Neurais Artificiais</b></h1>\n");
    if(usOpcao_Modo == 1)
        fprintf(fiArqOut1, "Modo de apresentação das figuras: Sedna</b></h1>\n");
    else if(usOpcao_Modo == 2)
        fprintf(fiArqOut1, "Modo de apresentação das figuras: Quaoar</b></h1>\n");
    else if(usOpcao_Modo == 3)
        fprintf(fiArqOut1, "Modo de apresentação das figuras: Xena</b></h1>\n");
    else if(usOpcao_Modo == 4)
        fprintf(fiArqOut1, "Modo de apresentação das figuras: Plutao</b></h1>\n");
    else if(usOpcao_Modo == 5)
        fprintf(fiArqOut1, "Modo de apresentação das figuras: Urano</b></h1>\n");
}

```

```

else if(usOpcao_Modo == 6)
    fprintf(fiArqOut1, "Modo de apresentação das figuras: Netuno</b></h1>\n");
else if(usOpcao_Modo == 7)
    fprintf(fiArqOut1, "Modo de apresentação das figuras: Saturno</b></h1>\n");
else if(usOpcao_Modo == 8)
    fprintf(fiArqOut1, "Modo de apresentação das figuras: Jupiter</b></h1>\n");
fprintf(fiArqOut1, "<p>\n");
}

void vGrava_Informacoes_Html(const unsigned short usOpcao_Dado, const unsigned short
usOpcao_Modo, const float fEta, const int iEpocas, const int iCont2, const int iCont3, const char *cNome1,
const char *cNome2, const char *cNome3, const char *cNome4, const char *cNome7, const char
*cNome8) {

    fprintf(fiArqOut1, "<h2 align = \"center\"><b>Informações gerais sobre o treinamento e
reconhecimento pela RNA </b></h2>\n");
    fprintf(fiArqOut1, "<table align = \"center\" border = 1>\n");
    fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Nome do Arquivo de Treinamento
Completo</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome1);
    fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Nome do Arquivo de
Treinamento</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome2);
    fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Nome do Arquivo de
Reconhecimento</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome3);
    fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Nome do Arquivo de Pesos
Sinápticos</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome4);
    fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Nome do Arquivo de
HiperTexto</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome7);
    fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Nome do Arquivo de Dados
Coletados</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome8);
    if(usOpcao_Dado == 1)
        fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Tipo de Imagem
Manipulada</h4>\n</td>\n<td>\n<h4><b>DIGITOS</b></h4>\n</td></th>\n");
    else
        fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Tipo de Imagem
Manipulada</h4>\n</td>\n<td>\n<h4><b>VOGAIS</b></h4>\n</td></th>\n");
    if(usOpcao_Modo == 1)
        fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Ordem de Apresentação dos Arquivos de Imagens
Manipulada</h4>\n</td>\n<td>\n<h4><b>SEDNA</b></h4>\n</td></th>\n");
    else if(usOpcao_Modo == 2)
        fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Ordem de Apresentação dos Arquivos de Imagens
Manipulada</h4>\n</td>\n<td>\n<h4><b>QUAOAR</b></h4>\n</td></th>\n");
    else if(usOpcao_Modo == 3)
        fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Ordem de Apresentação dos Arquivos de Imagens
Manipulada</h4>\n</td>\n<td>\n<h4><b>XENA</b></h4>\n</td></th>\n");
    else if(usOpcao_Modo == 4)
        fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Ordem de Apresentação dos Arquivos de Imagens
Manipulada</h4>\n</td>\n<td>\n<h4><b>PLUTAO</b></h4>\n</td></th>\n");
    else if(usOpcao_Modo == 5)
        fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Ordem de Apresentação dos Arquivos de Imagens
Manipulada</h4>\n</td>\n<td>\n<h4><b>URANO</b></h4>\n</td></th>\n");
    else if(usOpcao_Modo == 6)
        fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Ordem de Apresentação dos Arquivos de Imagens
Manipulada</h4>\n</td>\n<td>\n<h4><b>NETUNO</b></h4>\n</td></th>\n");
    else if(usOpcao_Modo == 7)

```

```

        fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Ordem de Apresentação dos Arquivos de Imagens
Manipulada</h4>\n</td>\n<td>\n<h4><b>SATURNO</b></h4>\n</td></th>\n");
        else if(usOpcao_Modo == 8)
            fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Ordem de Apresentação dos Arquivos de Imagens
Manipulada</h4>\n</td>\n<td>\n<h4><b>JUPITER</b></h4>\n</td></th>\n");
            fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Fator de Correção dos Pesos
Sinápticos</h4>\n</td>\n<td>\n<h4><b>%f</b></h4>\n</td></th>\n", fEta);
            fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Número de Epocas do
Treinamento</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iEpocas);
            fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Número de Arquivos de Imagens
Treinadas</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iCont2);
            fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Número de Arquivos de Imagens
Reconhecimento</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iCont3);
            fprintf(fiArqOut1, "</table>\n");
            fprintf(fiArqOut1, "<p>\n");
    }

```

```

void vGrava_Informacoes_Pesos_Html(const int iNCam, const int *iNNCam, const char *cNome4) {

```

```

    int il = 0;

    fprintf(fiArqOut1, "<h2 align = \"center\"><b>Informações sobre os pesos sináticos da RNA
</b></h2>\n");
    fprintf(fiArqOut1, "<table align = \"center\" border = 1>\n");
    fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Nome do Arquivo de Pesos
Sinápticos</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome4);
    fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Quantidade de Camadas da
RNA</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iNCam);
    fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Quantidade de Neurônios na Camada de
Entrada</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iNNCam[0]);
    for(il = 0; il < (iNCam - 2); il++)
        fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Quantidade de Neurônios na Camada Oculta
[%d]</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", il, iNNCam[il]);
    fprintf(fiArqOut1, "<tr>\n<td>\n<h4>Quantidade de Neurônios na Camada de
Entrada</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iNNCam[iNCam - 1]);
    fprintf(fiArqOut1, "</table>\n");
    fprintf(fiArqOut1, "<p>\n");
}

```

```

void vAnaliza_Conjunto_1(const int *iNNCam, const int iCont2, const unsigned short usOpcao_Dado) {

```

```

    int    il = 0,
           iJ = 0,
           iK = 0,
           iValor = 0,
           iVetorRespostaO[10][10] =

```

	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},



```

                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0});
char    cNome5[81],
        cInfo[2][10] = {{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'},
                        {'A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u'}};
float    dVetorResposta[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        dSoma = 0;

for(il = 0; il < iCont2; il++) {
    vLer_Nome_In_5(&iValor, cNome5);
    vLer_Imagem(iNNCam);
    for(iJ = 0; iJ < iNNCam[1]; iJ++) {
        dSoma = 0;
        for(iK = 0; iK < (iNNCam[0] + 1); iK++)
            dSoma += pdImagem[iK] * pdPeso[iJ][iK];
        dVetorResposta[iJ] = dSigmoide(dSoma);
    };
    iVetorRespostaO[iValor][iMaximo(dVetorResposta)]++;
    fclose(fiArqIn5);
};
fprintf(fiArqOut1, "<h2 align = \"center\"><b>Resultados Obtidos Para RNA sem Treinamento
usando o Arquivo de Treinamento</b></h2>\n");
fprintf(fiArqOut1, "<table align = \"center\" border = 1>\n");
fprintf(fiArqOut1, "<tr>\n<td rowspan = 2 align = center valign = middle>\n<h4>Valor
Desejado</h4>\n</td>\n");
fprintf(fiArqOut1, "<td colspan = 10 align = center valign = middle>\n<h4>Valor
Obtido</h4>\n</td>\n</tr>");
fprintf(fiArqOut1, "<tr>\n");
for(il = 0; il < 10; il++)
    fprintf(fiArqOut1, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
fprintf(fiArqOut1, "</tr>\n");
for(il = 0; il < 10; il++) {
    fprintf(fiArqOut1, "<tr>\n");
    fprintf(fiArqOut1, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
    for(iJ = 0; iJ < 10; iJ++)
        if(il == iJ)
            fprintf(fiArqOut1, "<td bgcolor = orange align = center valign = middle width
= 9%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[il][iJ]);
        else
            fprintf(fiArqOut1, "<td bgcolor = yellow text = white align = center valign =
middle width = 9%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[il][iJ]);
    fprintf(fiArqOut1, "</tr>\n");
};
fprintf(fiArqOut1, "</table>\n");
}

void vAnaliza_Conjunto_2(const int *iNNCam, const int iCont3, const unsigned short usOpcao_Dado) {
    int    il = 0,
        iJ = 0,
        iK = 0,
        iValor = 0,
        iVetorRespostaO[10][10] = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

```

```

                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};

char    cNome5[81],
        cInfo[2][10] = {{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'},
                          {'A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u'}};

float    dVetorResposta[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        dSoma = 0;

for(il = 0; il < iCont3; il++) {
    vLer_Nome_In_6(&iValor, cNome6);
    vLer_Imagem(iNNCam);
    for(iJ = 0; iJ < iNNCam[1]; iJ++) {
        dSoma = 0;
        for(iK = 0; iK < (iNNCam[0] + 1); iK++)
            dSoma += pdImagem[iK] * pdPeso[iJ][iK];
        dVetorResposta[iJ] = dSigmoid(dSoma);
    };
    iVetorRespostaO[iValor][iMaximo(dVetorResposta)]++;
    fclose(fiArqIn6);
};

fprintf(fiArqOut1, "<h2 align = \"center\"><b>Resultados Obtidos Para RNA sem Treinamento
usando o Arquivo de Reconhecimento</b></h2>\n");
fprintf(fiArqOut1, "<table align = \"center\" border = 1>\n");
fprintf(fiArqOut1, "<tr>\n<td rowspan = 2 align = center valign = middle>\n<h4>Valor
Desejado</h4>\n</td>\n");
fprintf(fiArqOut1, "<td colspan = 10 align = center valign = middle>\n<h4>Valor
Obtido</h4>\n</td>\n</tr>");
fprintf(fiArqOut1, "<tr>\n");
for(il = 0; il < 10; il++)
    fprintf(fiArqOut1, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
fprintf(fiArqOut1, "</tr>\n");
for(il = 0; il < 10; il++) {
    fprintf(fiArqOut1, "<tr>\n");
    fprintf(fiArqOut1, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
    for(iJ = 0; iJ < 10; iJ++)
        if(il == iJ)
            fprintf(fiArqOut1, "<td bgcolor = orange align = center valign = middle width
= 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[il][iJ]);
        else
            fprintf(fiArqOut1, "<td bgcolor = yellow text = white align = center valign =
middle width = 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[il][iJ]);
    fprintf(fiArqOut1, "</tr>\n");
};
fprintf(fiArqOut1, "</table>\n");
}

```

```

void vTreina_RNA(const int *iNNCam, const float fEta, const int iCont, const int iInter) {

    int    iCont2,
           il = 0,
           iJ = 0,
           iK = 0,
           iValor = 0;
    char    cNome5[81];
    float    dVetorRespostaD[10][10] =    {{1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                           {0, 1, 0, 0, 0, 0, 0, 0, 0, 0},
                                           {0, 0, 1, 0, 0, 0, 0, 0, 0, 0},
                                           {0, 0, 0, 1, 0, 0, 0, 0, 0, 0},
                                           {0, 0, 0, 0, 1, 0, 0, 0, 0, 0},
                                           {0, 0, 0, 0, 0, 1, 0, 0, 0, 0},
                                           {0, 0, 0, 0, 0, 0, 1, 0, 0, 0},
                                           {0, 0, 0, 0, 0, 0, 0, 1, 0, 0},
                                           {0, 0, 0, 0, 0, 0, 0, 0, 1, 0},
                                           {0, 0, 0, 0, 0, 0, 0, 0, 0, 1}},
           dVetorResposta[10] =           {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
           VetorErro[10] =                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
           dSoma = 0,
           dErro = 0;

    fscanf(fiArqIn2, "%d", &iCont2);
    for(il = 0; il < iCont2; il++) {
        vLer_Nome_In_5(&iValor, cNome5);
        vLer_Imagem(iNNCam);
        if((iCont == 1) || ((iCont % iInter) == 0))
            fprintf(fiArqOut2, "%s\t%d\t", cNome5, iValor);
        dErro = 0;
        for(iJ = 0; iJ < iNNCam[1]; iJ++) {
            dSoma = 0;
            for(iK = 0; iK < (iNNCam[0] + 1); iK++)
                dSoma += pdImagem[iK] * pdPeso[iJ][iK];
            dVetorResposta[iJ] = dSigmoide(dSoma);
            dVetorErro[iJ] = dVetorRespostaD[iValor][iJ] - dVetorResposta[iJ];
            if((iCont == 1) || ((iCont % iInter) == 0))
                fprintf(fiArqOut2, "%f\t", dVetorErro[iJ]);
            dErro += 0.5 * (dVetorRespostaD[iValor][iJ] - dVetorResposta[iJ]) *
(dVetorRespostaD[iValor][iJ] - dVetorResposta[iJ]);
        };
        if((iCont == 1) || ((iCont % iInter) == 0))
            fprintf(fiArqOut2, "\n");
        for(iK = 0; iK < (iNNCam[0] + 1); iK++)
            for(iJ = 0; iJ < iNNCam[1]; iJ++)
                pdPeso[iJ][iK] += fEta * dVetorErro[iJ] * (1 - dVetorResposta[iJ]) *
dVetorResposta[iJ] * pdImagem[iK];
        fclose(fiArqIn5);
    };
}

void vReconhe_RNA_T(const int *iNNCam, const int iEpocas, const unsigned short usOpcao_Dado) {

```

```

int    iCont2,
      il = 0,
      iJ = 0,
      iK = 0,
      iValor = 0,
      iVetorRespostaO[10][10] = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};

char   cNome5[81],
      cInfo[2][10] = {{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'},
                      {'A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u'}};

float  dVetorResposta[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
      dSoma = 0;

fscanf(fiArqIn2, "%d", &iCont2);
for(il = 0; il < iCont2; il++) {
    vLer_Nome_In_5(&iValor, cNome5);
    vLer_Imagem(iNNCam);
    for(iJ = 0; iJ < iNNCam[1]; iJ++) {
        dSoma = 0;
        for(iK = 0; iK < (iNNCam[0] + 1); iK++)
            dSoma += pdImagem[iK] * pdPeso[iJ][iK];
        dVetorResposta[iJ] = dSigmoide(dSoma);
    };
    iVetorRespostaO[iValor][iMaximo(dVetorResposta)]++;
    fclose(fiArqIn5);
};

fprintf(fiArqOut1, "<h2 align = \"center\"><b>Resultados Obtidos Para RNA Após o Treinamento
usando o Arquivo de Treinamento Época %d</b></h2>\n", iEpocas);
fprintf(fiArqOut1, "<table align = \"center\" border = 1>\n");
fprintf(fiArqOut1, "<tr>\n<td rowspan = 2 align = center valign = middle>\n<h4>Valor
Desejado</h4>\n</td>\n");
fprintf(fiArqOut1, "<td colspan = 10 align = center valign = middle>\n<h4>Valor
Obtido</h4>\n</td>\n</tr>");
fprintf(fiArqOut1, "<tr>\n");
for(il = 0; il < 10; il++)
    fprintf(fiArqOut1, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
fprintf(fiArqOut1, "</tr>\n");
for(il = 0; il < 10; il++) {
    fprintf(fiArqOut1, "<tr>\n");
    fprintf(fiArqOut1, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
    for(iJ = 0; iJ < 10; iJ++)
        if(il == iJ)
            fprintf(fiArqOut1, "<td bgcolor = orange align = center valign = middle width
= 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[iJ][iJ]);

```

```

        else
            fprintf(fiArqOut1, "<td bgcolor = yellow text = white align = center valign =
middle width = 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[iI][iJ]);
            fprintf(fiArqOut1, "</tr>\n");
        };
        fprintf(fiArqOut1, "</table>\n");
    }

void vReconhe_RNA_R(const int *iNNCam, const int iEpocas, const unsigned short usOpcao_Dado) {

    int    iCont3,
           iI = 0,
           iJ = 0,
           iK = 0,
           iValor = 0,
           iVetorRespostaO[10][10] = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}},

    char    cNome5[81],
           cInfo[2][10] = {{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'},
                           {'A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u'}};

    float    dVetorResposta[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
            dSoma = 0;

    fscanf(fiArqIn3, "%d", &iCont3);
    for(iI = 0; iI < iCont3; iI++) {
        vLer_Nome_In_6(&iValor, cNome6);
        vLer_Imagem(iNNCam);
        for(iJ = 0; iJ < iNNCam[1]; iJ++) {
            dSoma = 0;
            for(iK = 0; iK < (iNNCam[0] + 1); iK++)
                dSoma += pdImagem[iK] * pdPeso[iJ][iK];
            dVetorResposta[iJ] = dSigmoide(dSoma);
        };
        iVetorRespostaO[iValor][iMaximo(dVetorResposta)]++;
        fclose(fiArqIn6);
    };

    fprintf(fiArqOut1, "<h2 align = \"center\"><b>Resultados Obtidos Para RNA Após o Treinamento
usando o Arquivo de Reconhecimento Época %d</b></h2>\n", iEpocas);
    fprintf(fiArqOut1, "<table align = \"center\" border = 1>\n");
    fprintf(fiArqOut1, "<tr>\n<td rowspan = 2 align = center valign = middle>\n<h4>Valor
Desejado</h4>\n</td>\n");
    fprintf(fiArqOut1, "<td colspan = 10 align = center valign = middle>\n<h4>Valor
Obtido</h4>\n</td>\n</tr>");
    fprintf(fiArqOut1, "<tr>\n");
    for(iI = 0; iI < 10; iI++)

```

```

        fprintf(fiArqOut1, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
        fprintf(fiArqOut1, "</tr>\n");
        for(il = 0; il < 10; il++) {
            fprintf(fiArqOut1, "<tr>\n");
            fprintf(fiArqOut1, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
            for(iJ = 0; iJ < 10; iJ++)
                if(il == iJ)
                    fprintf(fiArqOut1, "<td bgcolor = orange align = center valign = middle width
= 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[iI][iJ]);
                else
                    fprintf(fiArqOut1, "<td bgcolor = yellow text = white align = center valign =
middle width = 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[iI][iJ]);
            fprintf(fiArqOut1, "</tr>\n");
        };
        fprintf(fiArqOut1, "</table>\n");
    }

void vGrava_Nome_RNA_T(const int *iNNCam, const unsigned short usOpcao_Dado) {

    int    iCont2,
           il = 0,
           iJ = 0,
           iK = 0,
           iValor = 0;

    char    cNome5[81],
           cInfo[2][10] = {{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'},
                           {'A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u'}};

    float    dVetorResposta[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
           dSoma = 0;

    fprintf(fiArqOut1, "<table align = \"center\" border = 1>\n");
    fprintf(fiArqOut1, "<tr>\n<td align = center valign = middle>\n<h4> </h4>\n</td>\n");
    fprintf(fiArqOut1, "<td colspan = 10 align = center valign = middle>\n<h4>Valor Obtido no
Treinamento</h4>\n</td>\n</tr>");
    fprintf(fiArqOut1, "<tr>\n<td align = center valign = middle>\n<h4>Nome do
Arquivo</h4>\n</td>\n");
    for(il = 0; il < 10; il++)
        fprintf(fiArqOut1, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
    fprintf(fiArqOut1, "</tr>\n");
    fscanf(fiArqIn2, "%d", &iCont2);
    for(il = 0; il < iCont2; il++) {
        vLer_Nome_In_5(&iValor, cNome5);
        vLer_Imagem(iNNCam);
        for(iJ = 0; iJ < iNNCam[1]; iJ++) {
            dSoma = 0;
            for(iK = 0; iK < (iNNCam[0] + 1); iK++)
                dSoma += pdImagem[iK] * pdPeso[iJ][iK];
            dVetorResposta[iJ] = dSigmoide(dSoma);
        };
        if(iValor != iMaximo(dVetorResposta)) {

```

```

        fprintf(fiArqOut1, "<tr>\n<td align = center valign =
middle>\n<h4>%s</h4>\n</td>\n", cNome5);
        for(iJ = 0; iJ < iNNCam[1]; iJ++)
            if(iJ == iValor)
                fprintf(fiArqOut1, "<td bgcolor = blue align = center valign
= middle>\n<h4>%f</h4>\n</td>\n", dVetorResposta[iJ]);
                else if(iJ == iMaximo(dVetorResposta))
                    fprintf(fiArqOut1, "<td bgcolor = red align = center valign =
middle>\n<h4>%f</h4>\n</td>\n", dVetorResposta[iJ]);
                else
                    fprintf(fiArqOut1, "<td bgcolor = yellow align = center
valign = middle>\n<h4>%f</h4>\n</td>\n", dVetorResposta[iJ]);
            };
        fprintf(fiArqOut1, "</tr>\n");
        fclose(fiArqIn5);
    };
    fprintf(fiArqOut1, "</table>\n");
}

void vGrava_Nome_RNA_R(const int *iNNCam, const unsigned short usOpcao_Dado) {

    int    iCont3,
           il = 0,
           iJ = 0,
           iK = 0,
           iValor = 0;
    char    cNome5[81],
           clnfo[2][10] = {{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'},
                           {'A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u'}};
    float    dVetorResposta[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
           dSoma = 0;

    fprintf(fiArqOut1, "<table align = \"center\" border = 1>\n");
    fprintf(fiArqOut1, "<tr>\n<td align = center valign = middle>\n<h4> </h4>\n</td>\n");
    fprintf(fiArqOut1, "<td colspan = 10 align = center valign = middle>\n<h4>Valor Obtido no
Treinamento</h4>\n</td>\n</tr>");
    fprintf(fiArqOut1, "<tr>\n<td align = center valign = middle>\n<h4>Nome do
Arquivo</h4>\n</td>\n");
    for(il = 0; il < 10; il++)
        fprintf(fiArqOut1, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
clnfo[usOpcao_Dado - 1][il]);
    fprintf(fiArqOut1, "</tr>\n");
    fscanf(fiArqIn3, "%d", &iCont3);
    for(il = 0; il < iCont3; il++) {
        vLer_Nome_In_6(&iValor, cNome6);
        vLer_Imagem(iNNCam);
        for(iJ = 0; iJ < iNNCam[1]; iJ++) {
            dSoma = 0;
            for(iK = 0; iK < (iNNCam[0] + 1); iK++)
                dSoma += pdImagem[iK] * pdPeso[iJ][iK];
            dVetorResposta[iJ] = dSigmoide(dSoma);
        };
        if(iValor != iMaximo(dVetorResposta)) {

```

```

        fprintf(fiArqOut1, "<tr>\n<td align = center valign =
middle>\n<h4>%s</h4>\n</td>\n", cNome6);
        for(iJ = 0; iJ < iNNCam[1]; iJ++)
            if(iJ == iValor)
                fprintf(fiArqOut1, "<td bgcolor = blue align = center valign
= middle>\n<h4>%lf</h4>\n</td>\n", dVetorResposta[iJ]);
                else if(iJ == iMaximo(dVetorResposta))
                    fprintf(fiArqOut1, "<td bgcolor = red align = center valign =
middle>\n<h4>%lf</h4>\n</td>\n", dVetorResposta[iJ]);
                else
                    fprintf(fiArqOut1, "<td bgcolor = yellow align = center
valign = middle>\n<h4>%lf</h4>\n</td>\n", dVetorResposta[iJ]);
            };
        fprintf(fiArqOut1, "</tr>\n");
        fclose(fiArqIn6);
    };
    fprintf(fiArqOut1, "</table>\n");
}

void vLer_Imagem(const int *iNNCam) {

    long int lil = 0;
    int      iPixelsX = 0,
            iPixelsY = 0;

    fread(&iPixelsX, sizeof(iPixelsX), 1, fiArqIn5);
    fread(&iPixelsY, sizeof(iPixelsY), 1, fiArqIn5);
    for(lil = 0; lil < (iNNCam[0] + 1); lil++)
        fread(&pdImagem[lil], sizeof(pdImagem[lil]), 1, fiArqIn5);
}

float dSigmoide(const float dSoma) {

    return(1 / (1 + exp(-dSoma)));
}

int iMaximo(const float *dVetorResposta) {

    int      il = 0,
            iAux = 0;
    float     dAux = 0;

    iAux = 0;
    dAux = dVetorResposta[iAux];
    for(il = 1; il < 10; il++)
        if(dAux < dVetorResposta[il]) {
            iAux = il;
            dAux = dVetorResposta[iAux];
        };
    return(iAux);
}

```



## A.6 Saturno1

```
#include <alloc.h>
#include <conio.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

char    cNome1[81],
        cNome2[81],
        cNome3[81],
        cNome4[81],
        cNome5[81],
        cNome6[81],
        cNome7[81];

float    huge    *pdPeso0[100],
        huge    *pdPeso1[10],
        huge    *pdImagem;
FILE    *fiArqIn1,
        *fiArqIn2,
        *fiArqIn3,
        *fiArqIn4,
        *fiArqIn5,
        *fiArqIn6,
        *fiArqOut;

void vApresentacao(void);

void vOtem_Tipo_de_Dado(unsigned short *usOpcao_Dado);

void vObtem_Nome_In_1(char *cNome1);

void vLer_Nome_In_2(char *cNome2);

void vLer_Nome_In_3(char *cNome3);

void vLer_Nome_In_4(char *cNome4);

void vLer_Nome_In_5(int *iValor, char *cNome5);

void vLer_Nome_In_6(int *iValor, char *cNome6);

void vLer_Nome_Out(char *cNome7);

void vAbrir_Arquivo_In_1(const char *cNome1);

void vAbrir_Arquivo_In_2(const char *cNome2);

void vAbrir_Arquivo_In_3(const char *cNome3);
```

```

void vCria_Espaco(const int *iNNCam, unsigned short *usOpcao_Inic);

void vLer_Peso(const int *iNNCam);

void vGravar_Peso(const int iNCam, const int *iNNCam);

void vGrava_Cabecalho_Html(const unsigned short usOpcao_Dado);

void vGrava_Informacoes_Html(const unsigned short usOpcao_Dado, const float fEta, const int iEpocas,
const int iCont2, const int iCont3, const char *cNome1, const char *cNome2, const char *cNome3, const
char *cNOme4, const char *cNome7);

void vGrava_Informacoes_Pesos_Html(const int iNCam, const int *iNNCam, const char *cNome4);

void vAnaliza_Conjunto_1(const int *iNNCam, const int iCont2, const unsigned short usOpcao_Dado);

void vAnaliza_Conjunto_2(const int *iNNCam, const int iCont3, const unsigned short usOpcao_Dado);

void vTreina_RNA(const int *iNNCam, const float fEta);

void vReconhe_RNA_T(const int *iNNCam, const int iEpocas, const unsigned short usOpcao_Dado);

void vReconhe_RNA_R(const int *iNNCam, const int iEpocas, const unsigned short usOpcao_Dado);

void vLer_Imagem(const int *iNNCam);

float dSigmoid(const float dSoma);

int iMaximo(const float *dVetorResposta);

void main(void) {
    int    iNCam = 0,
           iNNCam[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
           iCont2 = 0,
           iCont3 = 0,
           il = 0,
           iJ = 0,
           iEpocas = 0;
    float  fEta = 0;
    time_t  tLTInicio,
            tLTFim;
    unsigned short usCont = 0,
                  usOpcao_Inic = 0,
                  usOpcao_Dado = 0;

    vApresentacao();
    vOtem_Tipo_de_Dado(&usOpcao_Dado);
    vObtem_Nome_In_1(cNome1);
    vAbrir_Arquivo_In_1(cNome1);
    fscanf(fiArqIn1, "%u%f%d", &usCont, &fEta, &iEpocas);
    for(il = 0; il < usCont; il++) {
        tLTInicio = time(NULL);
        vLer_Nome_In_2(cNome2);
    }
}

```

```

vLer_Nome_In_3(cNome3);
vLer_Nome_In_4(cNome4);
vLer_Nome_Out(cNome7);
clrscr();
vApresentacao();
gotoxy(1,9);
if(usOpcao_Dado == 1)
    printf("Tipo de Dado: Digitos;");
else
    printf("Tipo de Dado: Vogais;");
gotoxy(1,10);
printf("Contador = %3d;", il + 1);
iJ = 0;
gotoxy(1,11);
printf("Epoca = %6d;", iJ);
fread(&iNCam, sizeof(iNCam), 1, fiArqIn4);
for(iJ = 0; iJ < iNCam; iJ++)
    fread(&iNNCam[iJ], sizeof(iNNCam[iJ]), 1, fiArqIn4);
vCria_Espaco(iNNCam, &usOpcao_Inic);
vLer_Peso(iNNCam);
vGrava_Cabecalho_Html(usOpcao_Dado);
fscanf(fiArqIn2,"%d",&iCont2);
fscanf(fiArqIn3,"%d",&iCont3);
vGrava_Informacoes_Html(usOpcao_Dado, fEta, iEpocas, iCont2, iCont3, cNome1,
cNome2, cNome3, cNome4, cNome7);
vGrava_Informacoes_Pesos_Html(iNCam, iNNCam, cNome4);
vAnaliza_Conjunto_1(iNNCam, iCont2, usOpcao_Dado);
vAnaliza_Conjunto_2(iNNCam, iCont3, usOpcao_Dado);
for(iJ = 1; iJ <= iEpocas; iJ++) {
    gotoxy(1,11);
    printf("Contador = %3d;", il + 1);
    gotoxy(1,12);
    printf("Epoca = %6d;", iJ);
    rewind(fiArqIn2);
    vTreina_RNA(iNNCam, fEta);
    if((iJ == (iEpocas / 4)) || (iJ == (iEpocas / 2)) || (iJ == (3 * iEpocas / 4)) || (iJ ==
iEpocas)) {
        rewind(fiArqIn2);
        rewind(fiArqIn3);
        vReconhe_RNA_T(iNNCam, iJ, usOpcao_Dado);
        vReconhe_RNA_R(iNNCam, iJ, usOpcao_Dado);
    };
};
vGravar_Peso(iNCam, iNNCam);
tLTFim = time(NULL);
fprintf(fiArqOut, "<table align = \"center\" border = 1>\n");
fprintf(fiArqOut, "<tr>\n<td>\n<h4>Data e Hora
Inicial</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", ctime(&tLTInicio));
fprintf(fiArqOut, "<tr>\n<td>\n<h4>Data e Hora
Inicial</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", ctime(&tLTFim));
fprintf(fiArqOut, "</table>\n");
fclose(fiArqIn2);
fclose(fiArqIn3);
fclose(fiArqIn4);

```

```

        fprintf(fiArqOut, "</body>\n</html>");
        fclose(fiArqOut);
    }
    fclose(fiArqIn1);
    farfree(pdImagem);
    farfree(pdPeso0);
    farfree(pdPeso1);
}

void vApresentacao(void) {

    clrscr();
    printf("Centro Universitario de Brasilia - UniCEUB\n");
    printf("Faculdade de Ciencias Exatas e Tecnologia - FAET\n");
}

void vOtem_Tipo_de_Dado(unsigned short *usOpcao_Dado) {

    printf("Entre com a opcao do tipo de dado a ser usado:\n1 = Digito.\n2 = Vogal.\nOpcao Dado = ");
    scanf("%u", usOpcao_Dado);
    if((*usOpcao_Dado != 1) && (*usOpcao_Dado != 2))
        *usOpcao_Dado = 1;
}

void vObtem_Tipo_de_Modo(unsigned short *usOpcao_Modo) {

    printf("\nEntre com a opcao do tipo do modo de treinamento e reconhecimento a ser usado:\n1 = Sedna.\n2 = Plutao.\nOpcao Modo = ");
    scanf("%u", usOpcao_Modo);
    if((*usOpcao_Modo != 1) && (*usOpcao_Modo != 2))
        *usOpcao_Modo = 1;
}

void vObtem_Nome_In_1(char *cNome1) {

    printf("\nEntre com o nome do arquivo completo de treinamento e reconhecimento da RNA:\nNome = ");
    scanf("%s", cNome1);
}

void vLer_Nome_In_2(char *cNome2) {

    fscanf(fiArqIn1, "%s", cNome2);
    fiArqIn2 = fopen(cNome2, "rt+");
    if(fiArqIn2 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome2);
        exit(1);
    };
}

void vLer_Nome_In_3(char *cNome3) {

    fscanf(fiArqIn1, "%s", cNome3);

```

```

        fiArqIn3 = fopen(cNome3, "rt+");
        if(fiArqIn3 == NULL) {
            printf("\nNão consegui abrir o arquivo: %s", cNome3);
            exit(1);
        };
    }

void vLer_Nome_In_4(char *cNome4) {

    fscanf(fiArqIn1, "%s", cNome4);
    fiArqIn4 = fopen(cNome4, "rb+");
    if(fiArqIn4 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome4);
        exit(1);
    };
}

void vLer_Nome_In_5(int *iValor, char *cNome5) {

    fscanf(fiArqIn2, "%d%s", iValor, cNome5);
    fiArqIn5 = fopen(cNome5, "rb");
    if(fiArqIn5 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome5);
        exit(1);
    };
}

void vLer_Nome_In_6(int *iValor, char *cNome6) {

    fscanf(fiArqIn3, "%d%s", iValor, cNome6);
    fiArqIn6 = fopen(cNome6, "rb");
    if(fiArqIn6 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome6);
        exit(1);
    };
}

void vLer_Nome_Out(char *cNome7) {

    fscanf(fiArqIn1, "%s", cNome7);
    fiArqOut = fopen(cNome7, "wt");
    if(fiArqOut == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome7);
        exit(1);
    };
}

void vAbrir_Arquivo_In_1(const char *cNome1) {

    fiArqIn1 = fopen(cNome1, "rt");
    if(fiArqIn1 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome1);
        exit(1);
    };
}

```

```

}

void vAbrir_Arquivo_In_2(const char *cNome2) {

    fiArqIn2 = fopen(cNome2, "rt");
    if(fiArqIn2 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome2);
        exit(1);
    };
}

void vAbrir_Arquivo_In_3(const char *cNome3) {

    fiArqIn3 = fopen(cNome3, "rt");
    if(fiArqIn3 == NULL) {
        printf("\nNão consegui abrir o arquivo: %s", cNome3);
        exit(1);
    };
}

void vCria_Espaco(const int *iNNCam, unsigned short *usOpcao_Inic) {

    int il;

    if(*usOpcao_Inic == 0) {
        pdImagem = (float huge *) farmalloc((iNNCam[0] + 1) * sizeof(float));
        for(il = 0; il < iNNCam[1]; il++)
            pdPeso0[il] = (float huge *) farmalloc((iNNCam[0] + 1) * sizeof(float)) ;
        for(il = 0; il < 10; il++)
            pdPeso1[il] = (float huge *) farmalloc((iNNCam[1] + 1) * sizeof(float)) ;
        *usOpcao_Inic = 1;
    };
}

void vLer_Peso(const int *iNNCam) {

    int    il = 0,
           iJ = 0;

    for(il = 0; il < iNNCam[1]; il++)
        for(iJ = 0; iJ < (iNNCam[0] + 1); iJ++)
            fread(&pdPeso0[il][iJ], sizeof(pdPeso0[il][iJ]), 1, fiArqIn4);
    for(il = 0; il < iNNCam[2]; il++)
        for(iJ = 0; iJ < (iNNCam[1] + 1); iJ++)
            fread(&pdPeso1[il][iJ], sizeof(pdPeso1[il][iJ]), 1, fiArqIn4);
}

void vGravar_Peso(const int iNCam, const int *iNNCam) {

    int    il = 0,
           iJ = 0;

    rewind(fiArqIn4);
    fwrite(&iNCam, sizeof(iNCam), 1, fiArqIn4);
}

```

```

        for(il = 0; il < iNNCam; il++)
            fwrite(&iNNCam[il], sizeof(iNNCam[il]), 1, fiArqIn4);
        for(il = 0; il < iNNCam[1]; il++)
            for(iJ = 0; iJ < (iNNCam[0] + 1); iJ++)
                fwrite(&pdPeso0[il][iJ], sizeof(pdPeso0[il][iJ]), 1, fiArqIn4);
        for(il = 0; il < iNNCam[2]; il++)
            for(iJ = 0; iJ < (iNNCam[1] + 1); iJ++)
                fwrite(&pdPeso1[il][iJ], sizeof(pdPeso1[il][iJ]), 1, fiArqIn4);
    }

void vGrava_Cabecalho_Html(const unsigned short usOpcao_Dado) {

    fprintf(fiArqOut, "<html>\n<head>\n");
    fprintf(fiArqOut, "<title>Treinamento e Reconhecimento de RNA com 0 camadas
ocultas</title>\n");
    fprintf(fiArqOut, "<\head><body bgcolor = \"white\" text = \"black\">\n");
    fprintf(fiArqOut, "<h1 align = \"center\"><b>Resultado do Treinamento e Reconhecimento de ");
    if(usOpcao_Dado == 1)
        fprintf(fiArqOut, "Dígitos pelas Redes Neurais Artificiais</b></h1>\n");
    else
        fprintf(fiArqOut, "Vogais pelas Redes Neurais Artificiais</b></h1>\n");
    fprintf(fiArqOut, "<p>\n");
}

void vGrava_Informacoes_Html(const unsigned short usOpcao_Dado, const float fEta, const int iEpocas,
const int iCont2, const int iCont3, const char *cNome1, const char *cNome2, const char *cNome3, const
char *cNome4, const char *cNome7) {

    fprintf(fiArqOut, "<h2 align = \"center\"><b>Informações gerais sobre o treinamento e
reconhecimento pela RNA </b></h2>\n");
    fprintf(fiArqOut, "<table align = \"center\" border = 1>\n");
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Nome do Arquivo de Treinamento
Completo</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome1);
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Nome do Arquivo de
Treinamento</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome2);
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Nome do Arquivo de
Reconhecimento</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome3);
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Nome do Arquivo de
Sinápticos</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome4);
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Nome do Arquivo de
HiperTexto</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome7);
    if(usOpcao_Dado == 1)
        fprintf(fiArqOut, "<tr>\n<td>\n<h4>Tipo de Imagem
Manipulada</h4>\n</td>\n<td>\n<h4><b>DIGITOS</b></h4>\n</td></th>\n");
    else
        fprintf(fiArqOut, "<tr>\n<td>\n<h4>Tipo de Imagem
Manipulada</h4>\n</td>\n<td>\n<h4><b>VOGAIS</b></h4>\n</td></th>\n");
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Fator de Correção dos Pesos
Sinápticos</h4>\n</td>\n<td>\n<h4><b>%f</b></h4>\n</td></th>\n", fEta);
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Número de Epocas do
Treinamento</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iEpocas);
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Número de Arquivos de Imagens
Treinadas</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iCont2);

```

```

        fprintf(fiArqOut, "<tr>\n<td>\n<h4>Número de Arquivos de Imagens
Reconhecimento</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iCont3);
        fprintf(fiArqOut, "</table>\n");
        fprintf(fiArqOut, "<p>\n");
    }

void vGrava_Informacoes_Pesos_Html(const int iNCam, const int *iNNCam, const char *cNome4) {

    int il = 0;

    fprintf(fiArqOut, "<h2 align = \"center\"><b>Informações sobre os pesos sináticos da RNA
</b></h2>\n");
    fprintf(fiArqOut, "<table align = \"center\" border = 1>\n");
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Nome do Arquivo de Pesos
Sinápticos</h4>\n</td>\n<td>\n<h4><b>%s</b></h4>\n</td></th>\n", cNome4);
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Quantidade de Camadas da
RNA</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iNCam);
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Quantidade de Neurônios na Camada de
Entrada</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iNNCam[0]);
    for(il = 0; il < (iNCam - 2); il++)
        fprintf(fiArqOut, "<tr>\n<td>\n<h4>Quantidade de Neurônios na Camada Oculta
[%d]</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", il, iNNCam[il]);
    fprintf(fiArqOut, "<tr>\n<td>\n<h4>Quantidade de Neurônios na Camada de
Entrada</h4>\n</td>\n<td>\n<h4><b>%d</b></h4>\n</td></th>\n", iNNCam[iNCam - 1]);
    fprintf(fiArqOut, "</table>\n");
    fprintf(fiArqOut, "<p>\n");
}

void vAnaliza_Conjunto_1(const int *iNNCam, const int iCont2, const unsigned short usOpcao_Dado) {

    int    il = 0,
           iJ = 0,
           iK = 0,
           iValor = 0,
           iVetorRespostaO[10][10] = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                       {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                       {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                       {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                       {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                       {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                       {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                       {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                       {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                       {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}},

    char    cNome5[81],
           cInfo[2][10] = {{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'},
                           {'A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u'}};

    float    dVetorResposta[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
            dSoma = 0,
            dTemp[100];

    for(il = 0; il < iCont2; il++) {
        vLer_Nome_In_5(&iValor, cNome5);
        vLer_Imagem(iNNCam);
    }
}

```





```

                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};

char    cNome5[81],
        cInfo[2][10] = {{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'},
                        {'A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u'}};

float    dVetorResposta[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        dSoma = 0,
        dTemp[100];

for(il = 0; il < iCont3; il++) {
    vLer_Nome_In_6(&iValor, cNome6);
    vLer_Imagem(iNNCam);
    for(iJ = 0; iJ < iNNCam[1]; iJ++) {
        dSoma = 0;
        for(iK = 0; iK < (iNNCam[0] + 1); iK++)
            dSoma += pdImagem[iK] * pdPeso0[iJ][iK];
        dTemp[iJ] = dSigmoide(dSoma);
    };
    for(iJ = 0; iJ < iNNCam[2]; iJ++) {
        dSoma = 0;
        for(iK = 0; iK < (iNNCam[1] + 1); iK++)
            dSoma += dTemp[iK] * pdPeso1[iJ][iK];
        dVetorResposta[iJ] = dSigmoide(dSoma);
    };
    iVetorRespostaO[iValor][iMaximo(dVetorResposta)]++;
    fclose(fiArqIn6);
};

fprintf(fiArqOut, "<h2 align = \"center\"><b>Resultados Obtidos Para RNA sem Treinamento
usando o Arquivo de Reconhecimento</b></h2>\n");
fprintf(fiArqOut, "<table align = \"center\" border = 1>\n");
fprintf(fiArqOut, "<tr>\n<td rowspan = 2 align = center valign = middle>\n<h4>Valor
Desejado</h4>\n</td>\n");
fprintf(fiArqOut, "<td colspan = 10 align = center valign = middle>\n<h4>Valor
Obtido</h4>\n</td>\n</tr>");
fprintf(fiArqOut, "<tr>\n");
for(il = 0; il < 10; il++)
    fprintf(fiArqOut, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
fprintf(fiArqOut, "</tr>\n");
for(il = 0; il < 10; il++) {
    fprintf(fiArqOut, "<tr>\n");
    fprintf(fiArqOut, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
    for(iJ = 0; iJ < 10; iJ++)
        if(il == iJ)
            fprintf(fiArqOut, "<td bgcolor = orange align = center valign = middle width
= 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[iJ][iJ]);
        else
            fprintf(fiArqOut, "<td bgcolor = yellow text = white align = center valign =
middle width = 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[iJ][iJ]);
    fprintf(fiArqOut, "</tr>\n");
}

```

```

    };
    fprintf(fiArqOut, "</table>\n");
}

void vTreina_RNA(const int *iNNCam, const float fEta) {

    int    iCont2,
           il = 0,
           iJ = 0,
           iK = 0,
           iValor = 0;
    char    cNome5[81];
    float    dVetorRespostaD[10][10] = {{1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                           {0, 1, 0, 0, 0, 0, 0, 0, 0, 0},
                                           {0, 0, 1, 0, 0, 0, 0, 0, 0, 0},
                                           {0, 0, 0, 1, 0, 0, 0, 0, 0, 0},
                                           {0, 0, 0, 0, 1, 0, 0, 0, 0, 0},
                                           {0, 0, 0, 0, 0, 1, 0, 0, 0, 0},
                                           {0, 0, 0, 0, 0, 0, 1, 0, 0, 0},
                                           {0, 0, 0, 0, 0, 0, 0, 1, 0, 0},
                                           {0, 0, 0, 0, 0, 0, 0, 0, 1, 0},
                                           {0, 0, 0, 0, 0, 0, 0, 0, 0, 1}},
           dVetorResposta[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
           dVetorErro[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
           dSoma = 0,
           dErro[100],
           dTemp[100];

    fscanf(fiArqIn2, "%d", &iCont2);
    for(il = 0; il < iCont2; il++) {
        vLer_Nome_In_5(&iValor, cNome5);
        vLer_Imagem(iNNCam);
        for(iJ = 0; iJ < iNNCam[1]; iJ++) {
            dSoma = 0;
            for(iK = 0; iK < (iNNCam[0] + 1); iK++)
                dSoma += pdImagem[iK] * pdPeso0[iJ][iK];
            dTemp[iJ] = dSigmoide(dSoma);
        };
        for(iJ = 0; iJ < iNNCam[2]; iJ++) {
            dSoma = 0;
            for(iK = 0; iK < (iNNCam[1] + 1); iK++)
                dSoma += dTemp[iK] * pdPeso1[iJ][iK];
            dVetorResposta[iJ] = dSigmoide(dSoma);
            dVetorErro[iJ] = dVetorRespostaD[iValor][iJ] - dVetorResposta[iJ];
        };
        for(iK = 0; iK < (iNNCam[1] + 1); iK++) {
            dSoma = 0;
            for(iJ = 0; iJ < iNNCam[2]; iJ++) {
                dSoma += dVetorErro[iJ] * pdPeso1[iJ][iK];
                pdPeso1[iJ][iK] += fEta * dVetorErro[iJ] * (1 - dVetorResposta[iJ]) *
dVetorResposta[iJ] * dTemp[iK];
            };
            dErro[iK] = dSoma;
        };
    };
}

```

```

        for(iK = 0; iK < (iNNCam[0] + 1); iK++)
            for(iJ = 0; iJ < iNNCam[1]; iJ++)
                pdPeso0[iJ][iK] += fEta * dErro[iJ] * (1 - dVetorResposta[iJ]) *
dVetorResposta[iJ] * pdImagem[iK];
        fclose(fiArqIn5);
    };
}

void vReconhe_RNA_T(const int *iNNCam, const int iEpocas, const unsigned short usOpcao_Dado) {

    int    iCont2,
           il = 0,
           iJ = 0,
           iK = 0,
           iValor = 0,
           iVetorRespostaO[10][10] = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}},

    char    cNome5[81],
           cInfo[2][10] = {{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'},
                           {'A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u'}};

    float    dVetorResposta[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
            dSoma = 0,
            dTemp[100];

    fscanf(fiArqIn2, "%d", &iCont2);
    for(il = 0; il < iCont2; il++) {
        vLer_Nome_In_5(&iValor, cNome5);
        vLer_Imagem(iNNCam);
        for(iJ = 0; iJ < iNNCam[1]; iJ++) {
            dSoma = 0;
            for(iK = 0; iK < (iNNCam[0] + 1); iK++)
                dSoma += pdImagem[iK] * pdPeso0[iJ][iK];
            dTemp[iJ] = dSigmoide(dSoma);
        };
        for(iJ = 0; iJ < iNNCam[2]; iJ++) {
            dSoma = 0;
            for(iK = 0; iK < (iNNCam[1] + 1); iK++)
                dSoma += dTemp[iK] * pdPeso1[iJ][iK];
            dVetorResposta[iJ] = dSigmoide(dSoma);
        };
        iVetorRespostaO[iValor][iMaximo(dVetorResposta)]++;
        fclose(fiArqIn5);
    };
    fprintf(fiArqOut, "<h2 align = \"center\"><b>Resultados Obtidos Para RNA Após o Treinamento
usando o Arquivo de Treinamento Época %d</b></h2>\n", iEpocas);
    fprintf(fiArqOut, "<table align = \"center\" border = 1>\n");
}

```

```

        fprintf(fiArqOut, "<tr>\n<td rowspan = 2 align = center valign = middle>\n<h4>Valor
Desejado</h4>\n</td>\n");
        fprintf(fiArqOut, "<td colspan = 10 align = center valign = middle>\n<h4>Valor
Obtido</h4>\n</td>\n</tr>");
        fprintf(fiArqOut, "<tr>\n");
        for(il = 0; il < 10; il++)
            fprintf(fiArqOut, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
        fprintf(fiArqOut, "</tr>\n");
        for(il = 0; il < 10; il++) {
            fprintf(fiArqOut, "<tr>\n");
            fprintf(fiArqOut, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
cInfo[usOpcao_Dado - 1][il]);
            for(iJ = 0; iJ < 10; iJ++)
                if(il == iJ)
                    fprintf(fiArqOut, "<td bgcolor = orange align = center valign = middle width
= 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[iJ][iJ]);
                else
                    fprintf(fiArqOut, "<td bgcolor = yellow text = white align = center valign =
middle width = 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[iJ][iJ]);
            fprintf(fiArqOut, "</tr>\n");
        };
        fprintf(fiArqOut, "</table>\n");
}

```

```

void vReconhe_RNA_R(const int *iNNCam, const int iEpocas, const unsigned short usOpcao_Dado) {

```

```

    int    iCont3,
           il = 0,
           iJ = 0,
           iK = 0,
           iValor = 0,
           iVetorRespostaO[10][10] = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}},

    char    cNome5[81],
           cInfo[2][10] = {{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'},
                           {'A', 'a', 'E', 'e', 'l', 'i', 'O', 'o', 'U', 'u'}};

    float    dVetorResposta[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
            dSoma = 0,
            dTemp[100];

    fscanf(fiArqIn3, "%d", &iCont3);
    for(il = 0; il < iCont3; il++) {
        vLer_Nome_In_6(&iValor, cNome6);
        vLer_Imagem(iNNCam);
        for(iJ = 0; iJ < iNNCam[1]; iJ++) {

```

```

        dSoma = 0;
        for(iK = 0; iK < (iNNCam[0] + 1); iK++)
            dSoma += pdImagem[iK] * pdPeso0[iJ][iK];
        dTemp[iJ] = dSigmoide(dSoma);
    };
    for(iJ = 0; iJ < iNNCam[2]; iJ++) {
        dSoma = 0;
        for(iK = 0; iK < (iNNCam[1] + 1); iK++)
            dSoma += dTemp[iK] * pdPeso1[iJ][iK];
        dVetorResposta[iJ] = dSigmoide(dSoma);
    };
    iVetorRespostaO[iValor][iMaximo(dVetorResposta)]++;
    fclose(fiArqIn6);
};
fprintf(fiArqOut, "<h2 align = \"center\"><b>Resultados Obtidos Para RNA Após o Treinamento
usando o Arquivo de Reconhecimento Época %d</b></h2>\n", iEpocas);
fprintf(fiArqOut, "<table align = \"center\" border = 1>\n");
fprintf(fiArqOut, "<tr>\n<td rowspan = 2 align = center valign = middle>\n<h4>Valor
Desejado</h4>\n</td>\n");
fprintf(fiArqOut, "<td colspan = 10 align = center valign = middle>\n<h4>Valor
Obtido</h4>\n</td>\n</tr>");
fprintf(fiArqOut, "<tr>\n");
for(il = 0; il < 10; il++)
    fprintf(fiArqOut, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
        clInfo[usOpcao_Dado - 1][il]);
fprintf(fiArqOut, "</tr>\n");
for(il = 0; il < 10; il++) {
    fprintf(fiArqOut, "<tr>\n");
    fprintf(fiArqOut, "<td align = center valign = middle>\n<h4><b>%c</b>\n<h4>\n</td>\n",
        clInfo[usOpcao_Dado - 1][il]);
    for(iJ = 0; iJ < 10; iJ++)
        if(il == iJ)
            fprintf(fiArqOut, "<td bgcolor = orange align = center valign = middle width
= 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[il][iJ]);
        else
            fprintf(fiArqOut, "<td bgcolor = yellow text = white align = center valign =
middle width = 9%%>\n<h4><b>%d</b>\n<h4>\n</td>\n", iVetorRespostaO[il][iJ]);
    fprintf(fiArqOut, "</tr>\n");
};
fprintf(fiArqOut, "</table>\n");
}

void vLer_Imagem(const int *iNNCam) {
    long int lil = 0;
    int iPixelsX = 0,
        iPixelsY = 0;

    fread(&iPixelsX, sizeof(iPixelsX), 1, fiArqIn5);
    fread(&iPixelsY, sizeof(iPixelsY), 1, fiArqIn5);
    for(lil = 0; lil < (iNNCam[0] + 1); lil++)
        fread(&pdImagem[lil], sizeof(pdImagem[lil]), 1, fiArqIn5);
}

```

```

float dSigmoide(const float dSoma) {
    return(1 / (1 + exp(-dSoma)));
}

int iMaximo(const float *dVetorResposta) {
    int    il = 0,
           iAux = 0;
    float   dAux = 0;

    iAux = 0;
    dAux = dVetorResposta[iAux];
    for(il = 1; il < 10; il++)
        if(dAux < dVetorResposta[il]) {
            iAux = il;
            dAux = dVetorResposta[iAux];
        };
    return(iAux);
}

```

## Anexo B – Resultados

Serão apresentados apenas alguns dos resultados, devido ao volume de dados ser bastante extenso. Estes resultados são uma copia autentica de como foram gerados os resultados do arquivo 'html'.

### Resultado do Treinamento e Reconhecimento de Vogais pelas Redes Neurais Artificiais

#### Informações gerais sobre o treinamento e reconhecimento pela RNA

<b>Nome do Arquivo de Treinamento Completo</b>	<b>SaturnVS.txt</b>
<b>Nome do Arquivo de Treinamento</b>	<b>VSedt04.txt</b>
<b>Nome do Arquivo de Reconhecimento</b>	<b>VSedr04.txt</b>
<b>Nome do Arquivo de Pesos Sinápticos</b>	<b>VSUra004.rna</b>
<b>Nome do Arquivo de HiperTexto</b>	<b>VSUra004.html</b>
<b>Tipo de Imagem Manipulada</b>	<b>VOGAIS</b>
<b>Ordem de Apresentação dos Arquivos de Imagens Manipulada</b>	<b>SEDNA</b>
<b>Fator de Correção dos Pesos Sinápticos</b>	<b>0.300000</b>
<b>Número de Epocas do Treinamento</b>	<b>5000</b>
<b>Número de Arquivos de Imagens Treinadas</b>	<b>1500</b>
<b>Número de Arquivos de Imagens Reconhecimento</b>	<b>500</b>

#### Informações sobre os pesos sináticos da RNA

<b>Nome do Arquivo de Pesos Sinápticos</b>	<b>VSUra004.rna</b>
--	---------------------



Quantidade de Camadas da RNA	2
Quantidade de Neurônios na Camada de Entrada	2500
Quantidade de Neurônios na Camada de Entrada	10

Resultados Obtidos Para RNA sem Treinamento usando o  
Arquivo de Treinamento

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	4	43	22	0	1	28	24	5	2	21
a	4	47	18	1	0	51	2	0	1	26
E	10	39	50	0	0	11	5	10	9	16
e	29	48	31	0	0	8	3	0	4	27
I	16	22	58	0	0	8	16	16	6	8
i	4	34	38	0	0	13	19	1	11	30
O	21	80	14	0	2	17	1	6	1	8
o	16	60	14	0	0	15	2	2	5	36
U	16	53	22	0	0	24	3	1	3	28
u	4	69	9	0	0	44	1	1	4	18

Resultados Obtidos Para RNA sem Treinamento usando o  
Arquivo de Reconhecimento

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	4	14	8	0	0	11	4	4	0	5

<b>a</b>	<b>1</b>	<b>14</b>	<b>5</b>	<b>0</b>	<b>0</b>	<b>17</b>	<b>1</b>	<b>2</b>	<b>1</b>	<b>9</b>
<b>E</b>	<b>4</b>	<b>15</b>	<b>17</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>3</b>	<b>2</b>	<b>5</b>
<b>e</b>	<b>12</b>	<b>15</b>	<b>7</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>3</b>	<b>7</b>
<b>I</b>	<b>1</b>	<b>10</b>	<b>15</b>	<b>0</b>	<b>0</b>	<b>5</b>	<b>12</b>	<b>4</b>	<b>3</b>	<b>0</b>
<b>i</b>	<b>0</b>	<b>13</b>	<b>11</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>5</b>	<b>0</b>	<b>5</b>	<b>14</b>
<b>O</b>	<b>10</b>	<b>23</b>	<b>6</b>	<b>0</b>	<b>1</b>	<b>3</b>	<b>4</b>	<b>0</b>	<b>2</b>	<b>1</b>
<b>o</b>	<b>5</b>	<b>20</b>	<b>5</b>	<b>0</b>	<b>0</b>	<b>5</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>13</b>
<b>U</b>	<b>0</b>	<b>19</b>	<b>4</b>	<b>0</b>	<b>0</b>	<b>11</b>	<b>0</b>	<b>0</b>	<b>4</b>	<b>12</b>
<b>u</b>	<b>2</b>	<b>20</b>	<b>4</b>	<b>0</b>	<b>0</b>	<b>17</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>6</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 1250

<b>Valor Desejado</b>	<b>Valor Obtido</b>									
	<b>A</b>	<b>a</b>	<b>E</b>	<b>e</b>	<b>I</b>	<b>i</b>	<b>O</b>	<b>o</b>	<b>U</b>	<b>u</b>
<b>A</b>	<b>140</b>	<b>4</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>0</b>
<b>a</b>	<b>0</b>	<b>140</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>2</b>
<b>E</b>	<b>0</b>	<b>0</b>	<b>145</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>
<b>e</b>	<b>3</b>	<b>0</b>	<b>1</b>	<b>132</b>	<b>2</b>	<b>6</b>	<b>1</b>	<b>4</b>	<b>0</b>	<b>1</b>
<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>149</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>145</b>	<b>2</b>	<b>2</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>144</b>	<b>1</b>	<b>3</b>	<b>0</b>
<b>o</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>4</b>	<b>143</b>	<b>1</b>	<b>1</b>
<b>U</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>141</b>	<b>5</b>

<b>u</b>	<b>5</b>	<b>3</b>	<b>1</b>	<b>3</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>2</b>	<b>133</b>
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	------------

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 1250

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
<b>A</b>	<b>37</b>	<b>4</b>	<b>2</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>3</b>
<b>a</b>	<b>7</b>	<b>33</b>	<b>0</b>	<b>3</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>4</b>	<b>0</b>	<b>0</b>
<b>E</b>	<b>3</b>	<b>0</b>	<b>39</b>	<b>1</b>	<b>2</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>0</b>	<b>0</b>
<b>e</b>	<b>3</b>	<b>0</b>	<b>0</b>	<b>34</b>	<b>1</b>	<b>3</b>	<b>2</b>	<b>5</b>	<b>1</b>	<b>1</b>
<b>I</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>44</b>	<b>4</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>40</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>3</b>
<b>O</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>35</b>	<b>1</b>	<b>8</b>	<b>1</b>
<b>o</b>	<b>0</b>	<b>3</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>41</b>	<b>1</b>	<b>0</b>
<b>U</b>	<b>0</b>	<b>4</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>2</b>	<b>39</b>	<b>2</b>
<b>u</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>40</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 2500

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
<b>A</b>	<b>142</b>	<b>2</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>0</b>
<b>a</b>	<b>0</b>	<b>140</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>1</b>
<b>E</b>	<b>2</b>	<b>0</b>	<b>145</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>

<b>e</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>132</b>	<b>2</b>	<b>5</b>	<b>1</b>	<b>4</b>	<b>1</b>	<b>2</b>
<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>149</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>145</b>	<b>2</b>	<b>2</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>144</b>	<b>1</b>	<b>1</b>	<b>1</b>
<b>o</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>143</b>	<b>2</b>	<b>0</b>
<b>U</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>142</b>	<b>3</b>
<b>u</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>3</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>3</b>	<b>135</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 2500

<b>Valor Desejado</b>	<b>Valor Obtido</b>									
	<b>A</b>	<b>a</b>	<b>E</b>	<b>e</b>	<b>I</b>	<b>i</b>	<b>O</b>	<b>o</b>	<b>U</b>	<b>u</b>
<b>A</b>	<b>37</b>	<b>4</b>	<b>2</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>3</b>
<b>a</b>	<b>7</b>	<b>32</b>	<b>0</b>	<b>3</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>3</b>	<b>1</b>	<b>1</b>
<b>E</b>	<b>3</b>	<b>0</b>	<b>39</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>0</b>
<b>e</b>	<b>4</b>	<b>0</b>	<b>0</b>	<b>32</b>	<b>1</b>	<b>3</b>	<b>2</b>	<b>5</b>	<b>1</b>	<b>2</b>
<b>I</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>45</b>	<b>3</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>39</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>2</b>
<b>O</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>34</b>	<b>2</b>	<b>8</b>	<b>0</b>
<b>o</b>	<b>0</b>	<b>3</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>41</b>	<b>1</b>	<b>0</b>
<b>U</b>	<b>2</b>	<b>4</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>36</b>	<b>3</b>
<b>u</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>39</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 3750

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	144	0	0	1	0	3	0	0	2	0
a	0	140	1	1	1	2	0	3	1	1
E	2	0	145	1	0	0	1	0	1	0
e	2	1	1	133	2	4	1	4	1	1
I	0	0	0	0	149	0	0	1	0	0
i	0	0	0	0	0	145	2	2	1	0
O	0	0	0	1	0	2	144	1	1	1
o	1	1	0	0	0	0	3	143	2	0
U	1	1	1	0	0	1	0	1	142	3
u	3	2	1	3	0	1	0	2	3	135

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 3750

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	38	3	2	0	1	1	1	0	1	3
a	7	33	0	2	0	2	1	3	1	1
E	3	0	40	1	2	2	2	0	0	0
e	3	0	0	32	1	3	2	5	2	2

<b>I</b>	0	1	0	0	44	3	1	0	0	1
<b>i</b>	2	1	1	0	3	38	0	2	1	2
<b>O</b>	2	1	0	2	1	0	33	2	9	0
<b>o</b>	0	3	0	2	0	1	2	41	1	0
<b>U</b>	2	4	0	1	1	0	2	2	35	3
<b>u</b>	1	2	2	1	0	2	0	2	1	39

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 5000

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
<b>A</b>	144	0	0	1	0	3	0	0	1	1
<b>a</b>	0	140	1	1	1	2	0	3	1	1
<b>E</b>	2	0	145	1	0	0	1	0	1	0
<b>e</b>	1	1	1	133	2	4	1	4	1	2
<b>I</b>	0	0	0	0	149	0	0	1	0	0
<b>i</b>	0	0	0	0	0	145	2	2	1	0
<b>O</b>	0	0	0	1	0	2	144	1	1	1
<b>o</b>	1	1	0	0	0	0	3	143	2	0
<b>U</b>	1	1	1	0	0	1	0	1	142	3
<b>u</b>	3	2	1	3	0	1	0	1	3	136

## Resultados Obtidos Para RNA Após o Treinamento usando o Arquivo de Reconhecimento Época 5000

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	37	3	2	0	1	1	2	0	1	3
a	7	33	0	2	0	2	1	3	1	1
E	3	0	40	1	2	2	2	0	0	0
e	3	0	0	32	1	3	2	5	2	2
I	0	1	0	0	45	3	1	0	0	0
i	1	1	1	0	3	39	0	2	1	2
O	3	1	0	1	1	0	32	3	9	0
o	0	3	0	2	0	1	2	41	1	0
U	2	3	0	1	1	0	2	3	35	3
u	2	1	2	1	0	2	0	2	1	39

Data e Hora Inicial	Mon Jul 10 05:34:26 2006
---------------------	--------------------------

Data e Hora Final	Mon Jul 10 23:26:18 2006
-------------------	--------------------------

Resultado do Treinamento e Reconhecimento de Vogais pelas Redes  
Neurais Artificiais

Informações gerais sobre o treinamento e reconhecimento pela  
RNA

Nome do Arquivo de Treinamento Completo	SaturnVP.txt
Nome do Arquivo de Treinamento	VPlut17.txt

<b>Nome do Arquivo de Reconhecimento</b>	<b>VPlur17.txt</b>
<b>Nome do Arquivo de Pesos Sinápticos</b>	<b>VPUra017.rna</b>
<b>Nome do Arquivo de HiperTexto</b>	<b>VPUra017.html</b>
<b>Tipo de Imagem Manipulada</b>	<b>VOGAIS</b>
<b>Ordem de Apresentação dos Arquivos de Imagens Manipulada</b>	<b>PLUTAO</b>
<b>Fator de Correção dos Pesos Sinápticos</b>	<b>0.300000</b>
<b>Número de Epocas do Treinamento</b>	<b>5000</b>
<b>Número de Arquivos de Imagens Treinadas</b>	<b>1500</b>
<b>Número de Arquivos de Imagens Reconhecimento</b>	<b>500</b>

### Informações sobre os pesos sináticos da RNA

<b>Nome do Arquivo de Pesos Sinápticos</b>	<b>VPUra017.rna</b>
<b>Quantidade de Camadas da RNA</b>	<b>2</b>
<b>Quantidade de Neurônios na Camada de Entrada</b>	<b>2500</b>
<b>Quantidade de Neurônios na Camada de Saída</b>	<b>10</b>

### Resultados Obtidos Para RNA sem Treinamento usando o Arquivo de Treinamento

<b>Valor Desejado</b>	<b>Valor Obtido</b>									
	<b>A</b>	<b>a</b>	<b>E</b>	<b>e</b>	<b>I</b>	<b>i</b>	<b>O</b>	<b>o</b>	<b>U</b>	<b>u</b>
<b>A</b>	<b>11</b>	<b>2</b>	<b>33</b>	<b>0</b>	<b>32</b>	<b>40</b>	<b>6</b>	<b>18</b>	<b>1</b>	<b>7</b>
<b>a</b>	<b>16</b>	<b>3</b>	<b>20</b>	<b>0</b>	<b>23</b>	<b>63</b>	<b>3</b>	<b>14</b>	<b>3</b>	<b>5</b>
<b>E</b>	<b>38</b>	<b>7</b>	<b>6</b>	<b>0</b>	<b>40</b>	<b>20</b>	<b>2</b>	<b>30</b>	<b>3</b>	<b>4</b>
<b>e</b>	<b>18</b>	<b>4</b>	<b>27</b>	<b>0</b>	<b>37</b>	<b>41</b>	<b>7</b>	<b>14</b>	<b>1</b>	<b>1</b>



<b>I</b>	14	7	15	0	16	53	8	19	0	18
<b>i</b>	5	0	26	0	20	55	8	9	3	24
<b>O</b>	31	33	6	0	22	19	5	21	5	8
<b>o</b>	14	5	32	0	23	56	3	16	0	1
<b>U</b>	17	18	20	2	30	23	9	16	10	5
<b>u</b>	19	7	33	1	28	28	2	28	2	2

Resultados Obtidos Para RNA sem Treinamento usando o  
Arquivo de Reconhecimento

<b>Valor Desejado</b>	<b>Valor Obtido</b>									
	<b>A</b>	<b>a</b>	<b>E</b>	<b>e</b>	<b>I</b>	<b>i</b>	<b>O</b>	<b>o</b>	<b>U</b>	<b>u</b>
<b>A</b>	3	0	10	0	7	14	3	8	1	4
<b>a</b>	3	2	7	0	12	16	1	4	3	2
<b>E</b>	6	3	1	0	18	6	2	11	1	2
<b>e</b>	4	2	4	0	11	21	3	5	0	0
<b>I</b>	2	2	3	0	6	19	3	2	0	13
<b>i</b>	2	0	6	0	10	15	1	6	2	8
<b>O</b>	7	13	6	0	2	5	1	7	5	4
<b>o</b>	8	1	3	0	10	21	1	4	2	0
<b>U</b>	7	2	9	0	9	9	4	6	2	2
<b>u</b>	8	2	6	0	11	9	1	12	1	0

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 1250

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	126	5	8	1	2	1	4	1	1	1
a	1	138	0	2	1	4	1	1	1	1
E	1	0	145	0	2	1	1	0	0	0
e	1	1	1	135	1	2	1	3	1	4
I	0	0	1	0	148	0	1	0	0	0
i	0	0	0	1	1	146	1	1	0	0
O	0	0	0	1	0	0	144	1	4	0
o	0	0	0	0	0	1	2	147	0	0
U	1	2	2	0	0	0	1	2	141	1
u	0	3	2	3	0	0	0	0	1	141

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 1250

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	35	3	1	0	1	3	3	0	2	2
a	2	37	0	3	0	1	1	1	2	3
E	2	0	42	0	3	0	1	2	0	0
e	2	2	2	36	0	1	1	3	1	2

<b>I</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>49</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>2</b>	<b>5</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>37</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>2</b>
<b>O</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>3</b>	<b>0</b>	<b>0</b>	<b>40</b>	<b>1</b>	<b>2</b>	<b>1</b>
<b>o</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>44</b>	<b>1</b>	<b>1</b>
<b>U</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>1</b>	<b>5</b>	<b>3</b>	<b>32</b>	<b>3</b>
<b>u</b>	<b>0</b>	<b>4</b>	<b>2</b>	<b>2</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>3</b>	<b>3</b>	<b>33</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 2500

<b>Valor Desejado</b>	<b>Valor Obtido</b>									
	<b>A</b>	<b>A</b>	<b>E</b>	<b>e</b>	<b>I</b>	<b>i</b>	<b>O</b>	<b>o</b>	<b>U</b>	<b>u</b>
<b>A</b>	<b>129</b>	<b>5</b>	<b>8</b>	<b>1</b>	<b>2</b>	<b>1</b>	<b>3</b>	<b>0</b>	<b>1</b>	<b>0</b>
<b>a</b>	<b>1</b>	<b>138</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>4</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>2</b>
<b>E</b>	<b>1</b>	<b>0</b>	<b>145</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>e</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>135</b>	<b>1</b>	<b>3</b>	<b>1</b>	<b>2</b>	<b>1</b>	<b>3</b>
<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>148</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>146</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>0</b>
<b>O</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>144</b>	<b>1</b>	<b>4</b>	<b>0</b>
<b>o</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>147</b>	<b>0</b>	<b>0</b>
<b>U</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>141</b>	<b>2</b>
<b>u</b>	<b>1</b>	<b>3</b>	<b>2</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>141</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 2500

Valor Desejado	Valor Obtido									
	A	A	E	e	I	i	O	o	U	u
A	36	3	1	0	1	3	3	0	2	1
a	2	38	0	3	0	1	0	1	2	3
E	2	0	42	0	3	0	1	2	0	0
e	2	2	3	37	0	1	1	1	1	2
I	2	0	0	0	48	0	0	0	0	0
i	2	5	1	0	2	37	0	0	1	2
O	1	0	3	3	0	0	40	0	2	1
o	0	2	0	1	0	2	0	42	1	2
U	0	2	1	1	2	1	5	3	33	2
u	0	5	2	2	0	2	1	3	3	32

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 3750

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	129	5	8	1	2	1	3	0	1	0
a	1	138	0	2	1	4	1	0	1	2
E	1	0	145	0	2	1	1	0	0	0
e	1	1	2	135	1	3	1	2	1	3

<b>I</b>	0	0	0	0	148	0	2	0	0	0
<b>i</b>	0	0	0	1	1	146	1	1	0	0
<b>O</b>	0	0	0	1	0	0	144	1	4	0
<b>o</b>	1	0	0	0	0	0	2	147	0	0
<b>U</b>	1	1	2	0	0	0	1	2	141	2
<b>u</b>	1	3	2	2	0	0	0	0	1	141

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 3750

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
<b>A</b>	35	3	1	0	2	3	3	0	2	1
<b>a</b>	2	38	0	3	0	1	0	1	2	3
<b>E</b>	2	0	42	0	3	0	1	2	0	0
<b>e</b>	2	2	3	37	0	1	1	1	1	2
<b>I</b>	1	0	0	0	49	0	0	0	0	0
<b>i</b>	2	5	1	0	2	37	0	0	1	2
<b>O</b>	1	0	3	3	0	0	40	0	2	1
<b>o</b>	0	2	0	1	0	2	0	42	1	2
<b>U</b>	0	2	1	1	2	1	6	3	32	2
<b>u</b>	0	6	2	2	0	2	1	3	3	31

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 5000

Valor Desejado	Valor Obtido									
	A	A	E	e	I	i	O	o	U	u
A	129	5	8	1	2	1	3	0	1	0
a	1	138	0	2	1	4	1	0	1	2
E	1	0	145	0	2	1	1	0	0	0
e	1	1	2	135	1	3	1	2	1	3
I	0	0	0	0	148	0	2	0	0	0
i	0	0	0	1	1	146	1	1	0	0
O	0	0	0	0	0	0	145	1	4	0
o	1	0	0	0	0	0	2	147	0	0
U	1	1	2	0	0	0	1	2	141	2
u	1	3	2	2	0	0	0	0	1	141

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 5000

Valor Desejado	Valor Obtido									
	A	A	E	e	I	i	O	o	U	u
A	34	3	1	0	2	3	3	0	2	2
a	2	38	0	3	0	1	0	1	2	3
E	2	0	42	0	3	0	1	2	0	0
e	3	2	3	36	0	1	1	1	1	2

<b>I</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>49</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>2</b>	<b>5</b>	<b>2</b>	<b>0</b>	<b>2</b>	<b>36</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>2</b>
<b>O</b>	<b>1</b>	<b>0</b>	<b>4</b>	<b>3</b>	<b>0</b>	<b>0</b>	<b>39</b>	<b>0</b>	<b>2</b>	<b>1</b>
<b>o</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>42</b>	<b>1</b>	<b>2</b>
<b>U</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>1</b>	<b>6</b>	<b>3</b>	<b>32</b>	<b>2</b>
<b>u</b>	<b>0</b>	<b>6</b>	<b>2</b>	<b>2</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>4</b>	<b>3</b>	<b>30</b>

<b>Data e Hora Inicial</b>	<b>Sat Aug 12 21:59:00 2006</b>
<b>Data e Hora Final</b>	<b>Sun Aug 13 15:51:10 2006</b>

Resultado do Treinamento e Reconhecimento de Vogais pelas Redes  
Neurais Artificiais

Informações gerais sobre o treinamento e reconhecimento pela  
RNA

<b>Nome do Arquivo de Treinamento Completo</b>	<b>CVenusV.txt</b>
<b>Nome do Arquivo de Treinamento</b>	<b>VSedt00.txt</b>
<b>Nome do Arquivo de Reconhecimento</b>	<b>VSedr00.txt</b>
<b>Nome do Arquivo de Pesos Sinápticos</b>	<b>VSed0001.rna</b>
<b>Nome do Arquivo de HiperTexto</b>	<b>VSed0001.html</b>
<b>Tipo de Imagem Manipulada</b>	<b>VOGAIS</b>
<b>Ordem de Apresentação dos Arquivos de Imagens Manipulada</b>	<b>SEDNA</b>
<b>Fator de Correção dos Pesos Sinápticos</b>	<b>0.300000</b>
<b>Número de Epocas do Treinamento</b>	<b>5000</b>

Número de Arquivos de Imagens Treinadas	2000
Número de Arquivos de Imagens Reconhecimento	2000

### Informações sobre os pesos sinápticos da RNA

Nome do Arquivo de Pesos Sinápticos	VSed0001.rna
Quantidade de Camadas da RNA	2
Quantidade de Neurônios na Camada de Entrada	2500
Quantidade de Neurônios na Camada de Saída	10

### Resultados Obtidos Para RNA sem Treinamento usando o Arquivo de Treinamento

Valor Desejado	Valor Obtido									
	A	A	E	e	I	i	O	o	U	u
A	3	9	0	30	0	0	2	5	90	61
a	0	2	0	54	0	4	0	6	62	72
E	1	2	1	60	1	2	3	33	67	30
e	0	0	0	54	0	3	0	34	63	46
I	3	13	0	62	2	1	7	16	64	32
i	2	0	0	76	12	1	1	4	61	43
O	1	8	1	36	0	1	8	34	63	48
o	0	0	0	68	1	3	1	5	31	91
U	5	3	0	62	2	2	3	16	53	54
u	0	0	0	106	1	3	0	7	23	60



Resultados Obtidos Para RNA sem Treinamento usando o  
Arquivo de Reconhecimento

Valor Desejado	Valor Obtido									
	A	A	E	e	I	i	O	o	U	u
A	3	9	0	30	0	0	2	5	90	61
a	0	2	0	54	0	4	0	6	62	72
E	1	2	1	60	1	2	3	33	67	30
e	0	0	0	54	0	3	0	34	63	46
I	3	13	0	62	2	1	7	16	64	32
i	2	0	0	76	12	1	1	4	61	43
O	1	8	1	36	0	1	8	34	63	48
o	0	0	0	68	1	3	1	5	31	91
U	5	3	0	62	2	2	3	16	53	54
u	0	0	0	106	1	3	0	7	23	60

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 1250

Valor Desejado	Valor Obtido									
	A	A	E	e	I	i	O	o	U	u
A	191	0	2	1	0	3	1	0	2	0
a	0	187	1	3	1	2	1	2	0	3
E	1	0	192	1	3	1	2	0	0	0
e	2	0	0	174	3	5	1	9	3	3

<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>200</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>192</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>193</b>	<b>1</b>	<b>3</b>	<b>1</b>
<b>o</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>3</b>	<b>192</b>	<b>2</b>	<b>1</b>
<b>U</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>2</b>	<b>189</b>	<b>3</b>
<b>u</b>	<b>4</b>	<b>4</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>4</b>	<b>185</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 1250

<b>Valor Desejado</b>	<b>Valor Obtido</b>									
	<b>A</b>	<b>A</b>	<b>E</b>	<b>e</b>	<b>I</b>	<b>i</b>	<b>O</b>	<b>o</b>	<b>U</b>	<b>u</b>
<b>A</b>	<b>191</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>0</b>
<b>a</b>	<b>0</b>	<b>187</b>	<b>1</b>	<b>3</b>	<b>1</b>	<b>2</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>3</b>
<b>E</b>	<b>1</b>	<b>0</b>	<b>192</b>	<b>1</b>	<b>3</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>e</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>174</b>	<b>3</b>	<b>5</b>	<b>1</b>	<b>9</b>	<b>3</b>	<b>3</b>
<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>200</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>192</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>193</b>	<b>1</b>	<b>3</b>	<b>1</b>
<b>o</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>3</b>	<b>192</b>	<b>2</b>	<b>1</b>
<b>U</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>2</b>	<b>189</b>	<b>3</b>
<b>u</b>	<b>4</b>	<b>4</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>4</b>	<b>185</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 2500

Valor Desejado	Valor Obtido									
	A	A	E	e	I	i	O	o	U	u
A	191	0	2	1	0	3	1	0	2	0
a	1	187	1	3	1	2	1	1	0	3
E	1	1	191	1	3	1	2	0	0	0
e	3	3	0	175	2	3	1	7	4	2
I	0	0	0	0	200	0	0	0	0	0
i	2	1	0	0	1	192	2	1	1	0
O	1	0	0	1	0	0	193	1	3	1
o	0	1	0	0	0	0	3	193	2	1
U	1	2	0	0	1	0	3	2	190	1
u	4	4	2	0	0	0	0	1	3	186

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 2500

Valor Desejado	Valor Obtido									
	A	A	E	e	I	i	O	o	U	u
A	191	0	2	1	0	3	1	0	2	0
a	1	187	1	3	1	2	1	1	0	3
E	1	1	191	1	3	1	2	0	0	0
e	3	3	0	175	2	3	1	7	4	2

<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>200</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>192</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>193</b>	<b>1</b>	<b>3</b>	<b>1</b>
<b>o</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>193</b>	<b>2</b>	<b>1</b>
<b>U</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>2</b>	<b>190</b>	<b>1</b>
<b>u</b>	<b>4</b>	<b>4</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>3</b>	<b>186</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 3750

<b>Valor Desejado</b>	<b>Valor Obtido</b>									
	<b>A</b>	<b>A</b>	<b>E</b>	<b>e</b>	<b>I</b>	<b>i</b>	<b>O</b>	<b>o</b>	<b>U</b>	<b>u</b>
<b>A</b>	<b>191</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>0</b>
<b>a</b>	<b>1</b>	<b>187</b>	<b>1</b>	<b>3</b>	<b>1</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>3</b>
<b>E</b>	<b>1</b>	<b>1</b>	<b>191</b>	<b>1</b>	<b>3</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>e</b>	<b>3</b>	<b>3</b>	<b>0</b>	<b>175</b>	<b>2</b>	<b>3</b>	<b>1</b>	<b>7</b>	<b>4</b>	<b>2</b>
<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>200</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>192</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>193</b>	<b>1</b>	<b>3</b>	<b>1</b>
<b>o</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>193</b>	<b>2</b>	<b>1</b>
<b>U</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>2</b>	<b>190</b>	<b>1</b>
<b>u</b>	<b>4</b>	<b>4</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>3</b>	<b>186</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 3750

Valor Desejado	Valor Obtido									
	A	A	E	e	I	i	O	o	U	u
A	191	0	2	1	0	2	1	0	3	0
a	1	187	1	3	1	2	1	1	0	3
E	1	1	191	1	3	1	2	0	0	0
e	3	3	0	175	2	3	1	7	4	2
I	0	0	0	0	200	0	0	0	0	0
i	2	1	0	0	1	192	2	1	1	0
O	1	0	0	1	0	0	193	1	3	1
o	0	1	0	0	0	0	3	193	2	1
U	1	2	0	0	1	0	3	2	190	1
u	4	4	2	0	0	0	0	1	3	186

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 5000

Valor Desejado	Valor Obtido									
	A	A	E	e	I	i	O	o	U	u
A	191	0	2	1	0	2	1	0	3	0
a	1	187	1	3	1	2	1	1	0	3
E	1	1	191	1	3	1	2	0	0	0
e	3	3	0	175	2	3	1	7	4	2

<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>200</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>192</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>193</b>	<b>1</b>	<b>3</b>	<b>1</b>
<b>o</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>193</b>	<b>2</b>	<b>0</b>
<b>U</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>2</b>	<b>190</b>	<b>1</b>
<b>u</b>	<b>4</b>	<b>4</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>3</b>	<b>186</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 5000

<b>Valor Desejado</b>	<b>Valor Obtido</b>									
	<b>A</b>	<b>A</b>	<b>E</b>	<b>e</b>	<b>I</b>	<b>i</b>	<b>O</b>	<b>o</b>	<b>U</b>	<b>u</b>
<b>A</b>	<b>191</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>0</b>
<b>a</b>	<b>1</b>	<b>187</b>	<b>1</b>	<b>3</b>	<b>1</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>3</b>
<b>E</b>	<b>1</b>	<b>1</b>	<b>191</b>	<b>1</b>	<b>3</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>e</b>	<b>3</b>	<b>3</b>	<b>0</b>	<b>175</b>	<b>2</b>	<b>3</b>	<b>1</b>	<b>7</b>	<b>4</b>	<b>2</b>
<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>200</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>192</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>193</b>	<b>1</b>	<b>3</b>	<b>1</b>
<b>o</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>193</b>	<b>2</b>	<b>0</b>
<b>U</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>3</b>	<b>2</b>	<b>190</b>	<b>1</b>
<b>u</b>	<b>4</b>	<b>4</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>3</b>	<b>186</b>

Data e Hora Inicial Tue Sep 26 19:06:09 2006

Data e Hora Final Wed Sep 27 18:59:53 2006

Resultado do Treinamento e Reconhecimento de Vogais pelas Redes  
Neurais Artificiais

Informações gerais sobre o treinamento e reconhecimento pela  
RNA

<b>Nome do Arquivo de Treinamento Completo</b>	<b>CVenusV.txt</b>
<b>Nome do Arquivo de Treinamento</b>	<b>VQuat00.txt</b>
<b>Nome do Arquivo de Reconhecimento</b>	<b>VQuar00.txt</b>
<b>Nome do Arquivo de Pesos Sinápticos</b>	<b>VQua0001.rna</b>
<b>Nome do Arquivo de HiperTexto</b>	<b>VQua0001.html</b>
<b>Tipo de Imagem Manipulada</b>	<b>VOGAIS</b>
<b>Ordem de Apresentação dos Arquivos de Imagens Manipulada</b>	<b>PLUTAO</b>
<b>Fator de Correção dos Pesos Sinápticos</b>	<b>0.450000</b>
<b>Número de Epocas do Treinamento</b>	<b>5000</b>
<b>Número de Arquivos de Imagens Treinadas</b>	<b>2000</b>
<b>Número de Arquivos de Imagens Reconhecimento</b>	<b>2000</b>

Informações sobre os pesos sináticos da RNA

<b>Nome do Arquivo de Pesos Sinápticos</b>	<b>VQua0001.rna</b>
<b>Quantidade de Camadas da RNA</b>	<b>2</b>
<b>Quantidade de Neurônios na Camada de Entrada</b>	<b>2500</b>
<b>Quantidade de Neurônios na Camada de Saída</b>	<b>10</b>

Resultados Obtidos Para RNA sem Treinamento usando o  
Arquivo de Treinamento

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	6	14	103	22	1	0	0	9	11	34
a	6	44	71	14	2	0	0	21	9	33
E	4	53	76	19	10	4	2	4	8	20
e	3	28	134	2	0	0	0	3	2	28
I	3	24	96	22	0	4	2	0	18	31
i	2	16	125	10	1	8	2	2	7	27
O	5	40	88	20	5	0	0	9	4	29
o	2	20	128	12	1	3	0	8	1	25
U	8	15	95	28	7	2	1	5	5	34
u	8	21	106	19	1	3	1	5	7	29

Resultados Obtidos Para RNA sem Treinamento usando o  
Arquivo de Reconhecimento

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	6	14	103	22	1	0	0	9	11	34
a	6	44	71	14	2	0	0	21	9	33
E	4	53	76	19	10	4	2	4	8	20
e	3	28	134	2	0	0	0	3	2	28



<b>I</b>	3	24	96	22	0	4	2	0	18	31
<b>i</b>	2	16	125	10	1	8	2	2	7	27
<b>O</b>	5	40	88	20	5	0	0	9	4	29
<b>o</b>	2	20	128	12	1	3	0	8	1	25
<b>U</b>	8	15	95	28	7	2	1	5	5	34
<b>u</b>	8	21	106	19	1	3	1	5	7	29

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 1250

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
<b>A</b>	189	1	2	1	0	2	3	0	2	0
<b>a</b>	0	189	0	3	1	0	0	1	0	6
<b>E</b>	1	0	192	1	2	1	2	0	1	0
<b>e</b>	1	1	0	174	3	4	2	9	2	4
<b>I</b>	0	0	0	0	200	0	0	0	0	0
<b>i</b>	3	1	0	0	1	191	2	1	1	0
<b>O</b>	0	0	0	0	0	1	195	0	3	1
<b>o</b>	1	2	0	0	0	1	2	191	3	0
<b>U</b>	0	2	0	0	1	0	1	1	192	3
<b>u</b>	2	2	3	0	0	0	0	1	1	191

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 1250

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	189	1	2	1	0	2	3	0	2	0
a	0	189	0	3	1	0	0	1	0	6
E	1	0	192	1	2	1	2	0	1	0
e	1	1	0	174	3	4	2	9	2	4
I	0	0	0	0	200	0	0	0	0	0
i	3	1	0	0	1	191	2	1	1	0
O	0	0	0	0	0	1	195	0	3	1
o	1	2	0	0	0	1	2	191	3	0
U	0	2	0	0	1	0	1	1	192	3
u	2	2	3	0	0	0	0	1	1	191

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 2500

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	190	0	2	2	0	2	2	0	2	0
a	0	190	1	3	1	1	0	1	0	3
E	1	0	192	1	2	1	2	0	1	0
e	2	2	0	176	3	4	2	7	2	2

<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>200</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>3</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>191</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>195</b>	<b>0</b>	<b>4</b>	<b>0</b>
<b>o</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>192</b>	<b>2</b>	<b>0</b>
<b>U</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>193</b>	<b>3</b>
<b>u</b>	<b>3</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>3</b>	<b>190</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 2500

<b>Valor Desejado</b>	<b>Valor Obtido</b>									
	<b>A</b>	<b>a</b>	<b>E</b>	<b>e</b>	<b>I</b>	<b>i</b>	<b>O</b>	<b>o</b>	<b>U</b>	<b>u</b>
<b>A</b>	<b>190</b>	<b>0</b>	<b>2</b>	<b>2</b>	<b>0</b>	<b>2</b>	<b>2</b>	<b>0</b>	<b>2</b>	<b>0</b>
<b>a</b>	<b>0</b>	<b>190</b>	<b>1</b>	<b>3</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>3</b>
<b>E</b>	<b>1</b>	<b>0</b>	<b>192</b>	<b>1</b>	<b>2</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>1</b>	<b>0</b>
<b>e</b>	<b>2</b>	<b>2</b>	<b>0</b>	<b>176</b>	<b>3</b>	<b>4</b>	<b>2</b>	<b>7</b>	<b>2</b>	<b>2</b>
<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>200</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>3</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>191</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>195</b>	<b>0</b>	<b>4</b>	<b>0</b>
<b>o</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>192</b>	<b>2</b>	<b>0</b>
<b>U</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>193</b>	<b>3</b>
<b>u</b>	<b>3</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>3</b>	<b>190</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 3750

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	190	0	2	1	0	2	3	0	2	0
a	1	190	1	3	1	1	0	1	0	2
E	1	0	192	1	2	1	2	0	1	0
e	2	2	0	176	3	4	2	7	2	2
I	0	0	0	0	200	0	0	0	0	0
i	3	2	0	0	0	191	2	1	1	0
O	0	0	0	0	1	1	194	0	4	0
o	1	2	0	0	0	0	3	192	2	0
U	0	1	0	0	1	0	1	1	193	3
u	4	0	2	0	0	0	0	1	3	190

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 3750

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	190	0	2	1	0	2	3	0	2	0
a	1	190	1	3	1	1	0	1	0	2
E	1	0	192	1	2	1	2	0	1	0
e	2	2	0	176	3	4	2	7	2	2

<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>200</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>3</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>191</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>194</b>	<b>0</b>	<b>4</b>	<b>0</b>
<b>o</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>192</b>	<b>2</b>	<b>0</b>
<b>U</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>193</b>	<b>3</b>
<b>u</b>	<b>4</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>3</b>	<b>190</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Treinamento Época 5000

<b>Valor Desejado</b>	<b>Valor Obtido</b>									
	<b>A</b>	<b>a</b>	<b>E</b>	<b>e</b>	<b>I</b>	<b>i</b>	<b>O</b>	<b>o</b>	<b>U</b>	<b>u</b>
<b>A</b>	<b>190</b>	<b>0</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>3</b>	<b>0</b>	<b>2</b>	<b>0</b>
<b>a</b>	<b>1</b>	<b>190</b>	<b>1</b>	<b>3</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>2</b>
<b>E</b>	<b>1</b>	<b>0</b>	<b>192</b>	<b>1</b>	<b>2</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>1</b>	<b>0</b>
<b>e</b>	<b>2</b>	<b>2</b>	<b>0</b>	<b>176</b>	<b>3</b>	<b>4</b>	<b>2</b>	<b>7</b>	<b>2</b>	<b>2</b>
<b>I</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>200</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>i</b>	<b>3</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>191</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>O</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>194</b>	<b>0</b>	<b>4</b>	<b>0</b>
<b>o</b>	<b>1</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>3</b>	<b>192</b>	<b>2</b>	<b>0</b>
<b>U</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>193</b>	<b>3</b>
<b>u</b>	<b>4</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>3</b>	<b>190</b>

Resultados Obtidos Para RNA Após o Treinamento usando o  
Arquivo de Reconhecimento Época 5000

Valor Desejado	Valor Obtido									
	A	a	E	e	I	i	O	o	U	u
A	190	0	2	1	0	2	3	0	2	0
a	1	190	1	3	1	1	0	1	0	2
E	1	0	192	1	2	1	2	0	1	0
e	2	2	0	176	3	4	2	7	2	2
I	0	0	0	0	200	0	0	0	0	0
i	3	2	0	0	0	191	2	1	1	0
O	0	0	0	0	1	1	194	0	4	0
o	1	2	0	0	0	0	3	192	2	0
U	0	1	0	0	1	0	1	1	193	3
u	4	0	2	0	0	0	0	1	3	190

Data e Hora Inicial	Fri Oct 13 13:40:15 2006
Data e Hora Final	Sun Oct 15 09:17:27 2006